

---

# Projet N°227 : Caravel

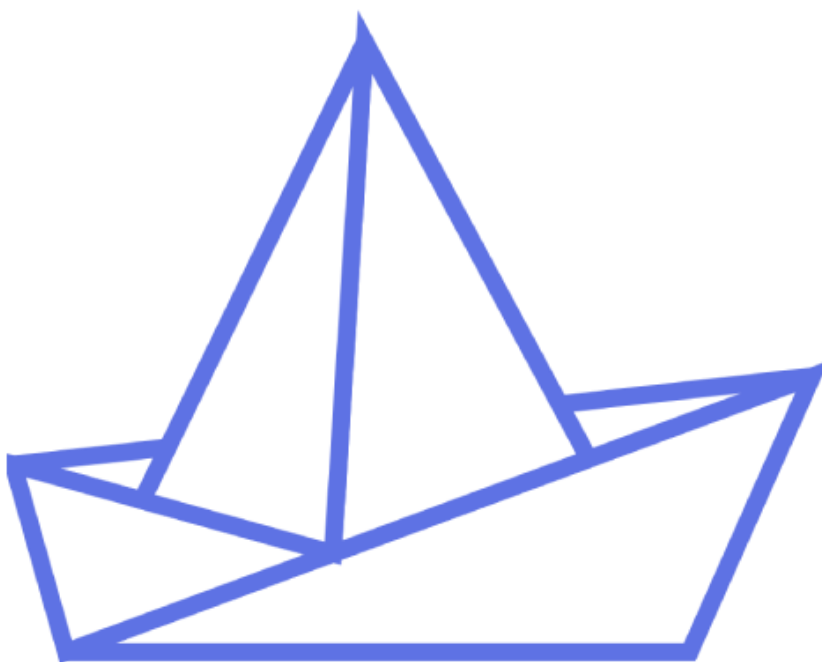
Rapport

Travail de Bachelor

A l'attention de MM. XXX

Mendes Reis Steve

21 juillet 2021



## Table des matières

0.1	Résumé . . . . .	5
0.2	abstract . . . . .	5
<b>1</b>	<b>Introduction</b>	<b>6</b>
1.1	Contexte de développement . . . . .	6
<b>2</b>	<b>Cahier des charges</b>	<b>7</b>
2.1	Situation initiale . . . . .	7
2.2	Buts visés . . . . .	7
2.2.1	Contraintes éventuelles . . . . .	8
2.2.2	Priorisation des tâches . . . . .	8
<b>3</b>	<b>Analyse</b>	<b>10</b>
3.1	Problématique . . . . .	10
3.1.1	Pour les professeurs . . . . .	10
3.1.2	Pour les élèves . . . . .	10
3.2	Etat de l'art . . . . .	11
3.2.1	Applications testées . . . . .	11
3.2.2	MyStudyLife . . . . .	14
3.2.3	Synthèse . . . . .	16
3.2.4	Positionnement de Caravel . . . . .	16
3.3	Conception . . . . .	17
3.3.1	Rôles et use cases . . . . .	17
3.3.2	Use cases . . . . .	18
3.3.3	Gestion des filtres . . . . .	23
3.3.4	Système de réactions . . . . .	24
3.3.5	Système de notifications . . . . .	25
3.3.6	Gestion de la charge de travail . . . . .	25
3.3.7	Modélisation de la base de données . . . . .	28
3.3.8	Système d'authentification . . . . .	29
3.4	Définitions des routes . . . . .	29
3.5	Stratégie & conception de test . . . . .	29
3.5.1	Attentes de la qualité du produit . . . . .	29
3.5.2	Objectifs de tests . . . . .	29
3.5.3	Périmètre de tests . . . . .	29
3.5.4	Gestion des risques . . . . .	29
3.6	Maquettes . . . . .	30
3.7	Planning . . . . .	35
3.8	Méthodologie de travail . . . . .	35

<b>4</b>	<b>Implémentation</b>	<b>37</b>
4.1	Choix de la base de donnée . . . . .	37
4.2	Authentification . . . . .	37
4.2.1	Local Storage vs Cookies . . . . .	37
4.2.2	Sanctum vs Passport . . . . .	38
4.2.3	LDAP . . . . .	39
4.3	DevOps CI/CD . . . . .	39
4.3.1	Intégration continue . . . . .	39
4.3.2	Livraison continue . . . . .	40
4.3.3	Environnement de production . . . . .	42
4.4	Système de notification . . . . .	43
4.4.1	Déclenchement d'une notification . . . . .	44
4.4.2	Configuration de FCM . . . . .	46
4.4.3	Récupération des notifications depuis le frontend . . . . .	46
4.5	Frontend . . . . .	46
4.5.1	Vuex . . . . .	46
4.5.2	Localisation . . . . .	46
4.5.3	Composants . . . . .	46
4.5.4	Filtres . . . . .	47
4.5.5	Gestion des erreurs Axios . . . . .	47
4.5.6	Gestion du chargement . . . . .	47
4.6	Backend . . . . .	47
4.6.1	Policies . . . . .	47
4.6.2	Validation des requêtes . . . . .	47
4.6.3	Localisation . . . . .	47
4.6.4	Moteur de recherche . . . . .	47
4.6.5	Gestion de clés locales . . . . .	47
<b>5</b>	<b>User tests</b>	<b>47</b>
5.1	Scénario . . . . .	47
5.2	Résultats . . . . .	48
<b>6</b>	<b>Améliorations</b>	<b>49</b>
<b>7</b>	<b>Conclusion</b>	<b>50</b>
<b>8</b>	<b>Annexes</b>	<b>51</b>

## Table des figures

1	Google Classroom : Liste des types de devoirs . . . . .	11
2	Google Classroom : Gestion de liste d'élèves . . . . .	12
3	Google Classroom : Permet de faire des annonces . . . . .	12
4	Google Classroom : Vue limité à la semaine . . . . .	12
5	MyHomework : Ajout d'un devoir . . . . .	13
6	MyHomework : Vue centrale simple et efficace . . . . .	13
7	MyHomework : Vue calendrier (mensuelle) . . . . .	14
8	MyHomework : Vue semaine . . . . .	14
9	MyStudyLife : Gestion des tâches . . . . .	15
10	MyStudyLife : Gestion des différents sujets . . . . .	15
11	MyStudyLife : Gestion des cours . . . . .	16
12	Use case : gestion des tâches . . . . .	18
13	Use case : gestion des sujets . . . . .	19
14	Use case : gestion des fils de discussion . . . . .	20
15	Use case : gestion des notifications . . . . .	21
16	Use case : gestion des filtres . . . . .	22
17	Use case : gestion du login . . . . .	23
18	Maquette des paramètres de notifications . . . . .	25
19	Modélisation de la base de données . . . . .	28
20	Maquette : page de login . . . . .	31
21	Maquette : page liste des tâches . . . . .	32
22	Maquette : page affichage d'une tâche . . . . .	33
23	Maquette : page vue mensuelle . . . . .	34
24	Maquette : page vue timeline . . . . .	34
25	Maquette : page de statistiques . . . . .	35
26	GitFlow workflow feature branches . . . . .	36
27	Schéma d'interaction pour l'authentification . . . . .	37
28	Laravel Sanctum Explained : SPA Authentication Bailly (2020) . . . . .	38
29	Schéma global du système de notifications . . . . .	43

## Liste des codes

1	Fonction de login . . . . .	38
2	Pipeline de test Laravel SQLite . . . . .	39
3	Serveur : configuration Nginx . . . . .	41
4	Notifications : canaux de diffusion . . . . .	44
5	Notification : comportement des canaux . . . . .	45
6	Notification : envoi asynchrone . . . . .	45

## 0.1 Résumé

## 0.2 abstract

# 1 Introduction

## 1.1 Contexte de développement

- Visual Code
- Docker

## 2 Cahier des charges

Cette section décrit le cadre initial du projet ainsi que ses buts visés à la fin du développement

### 2.1 Situation initiale

Ce projet fait suite à un travail réalisé par trois étudiants dans le cadre du cours de Développement Web. Dans le cadre de ce projet les trois étudiants ont réalisé une plateforme Web permettant aux élèves d'inscrire de manière collaborative les différents tâches (devoirs/CP/Projet) à faire.

### 2.2 Buts visés

Avec Caravel l'idée est que les devoirs soient gérés par les membres d'une classe, chaque membre de la classe a donc la possibilité de renseigner un devoir sur la plateforme. Si celui-ci manque de précision, le professeur ou un élève peut y apporter une modification en tout temps afin d'éclaircir son contenu. Dans cette optique autant les professeurs que les élèves peuvent être une source de devoirs ce qui change la dynamique des élèves par rapport aux applications habituelles.

Un des but de Caravel est de réunir tous les devoirs en un seul endroit, avec l'apparition des différents outils comme Moodle, Teams, l'intranet ou encore les dossiers partagés, il est parfois très difficile de savoir où chercher l'information concernant un devoir. L'idée est donc de permettre aux membres d'une classe de réunir les informations pertinentes en un seul endroit afin de gagner du temps. La contribution collaborative permet de faciliter la transition à l'utilisation de Caravel, si par exemple un prof continue de distribuer ses exercices sous format PDF par des canaux différents, il suffit qu'un seul élève soit au courant de l'information pour la transmettre aux autres en les ajoutant sur Caravel.

De plus Caravel pourra permettre d'ouvrir des fils de discussion au sein d'une tâche afin de demander de l'aide ou des éclaircissement sur une tâche, l'idée est de permettre aux élèves de s'entraider et de partager en un seul lieu les différentes informations (questions/réponses).

Enfin, l'outil devra permettre d'avoir une vue de la charge de travail d'une classe afin de placer au mieux les prochains CP/devoirs, également il sera possible pour un élève de réagir sur les différentes tâches pour alerter les professeurs sur un devoir qui serait inadapté (temps de travail, complexité, manque d'informations, etc...)

Succinctement les buts sont les suivants :

- Placer au mieux les devoirs et CP pour lisser la charge et savoir si la charge de travail est correcte
- Réunir en un seul endroit toutes les informations relatives au travail qui doit être effectuer par l'étudiant et ainsi éviter l'utilisation de différents canaux
- Aider les élèves à mieux s'organiser en ayant une place qui réuni toutes les informations nécessaires à l'exécution correcte de leur travail
- Proposer une plateforme de collaboration entre étudiants et professeurs sur des tâches via des échanges questions/réponses



### 2.2.1 Contraintes éventuelles

Pas de contraintes

### 2.2.2 Priorisation des tâches

Une priorisation des tâches a été effectuée afin de déterminer les éléments importants du projet. Cette priorisation est basée sur la *méthode de MoSCoW* (2021).

#### 2.2.2.1 Must have

- Traduction en français
  - Prévoir la possibilité d'ajouter d'autres langues facilement (localisation)
- Filtres sur les tâches dans les différentes vues
  - Filtres par titre, par sujet, par auteur
- Rôles étudiant/professeur
  - Les profs pourront ajouter des tâches, elles ne seront alors pas modifiables par les élèves
- Ajout d'une vue en mois style Outlook
  - Permet de naviguer sur d'anciennes semaines
- Séparation front/back end
  - Passage en Vue.js, Laravel en backend
- Analyse & intégration des feedbacks reçus pendant le semestre de printemps
- Réactions aux différentes tâches
  - Pouvoir réagir une tâche (trop long, trop complexe, etc...)
- Ajout de notion de crédit sur les sujets

#### 2.2.2.2 Should have

- Système de notifications
  - par ex. une tâche a été ajoutée ou modifiée, réponse à un commentaire
- Authentification interne (LDAP)
- Système de sujets(fil) dans les commentaires au niveau des tâches
  - Possibilité d'éditer les commentaires
- Ajout de la représentation de la charge de travail

#### 2.2.2.3 Could have

- PWA
  - Possibilité de notification push
- Possibilité pour un élève de mettre une tâche comme terminée pour lui uniquement
- Partage des tâches privées avec certains membres
- Enrichir l'éditeur de texte
  - ajouter des mentions type @membre, #idTache

- Gestion des paramètres de notifications
- Onglet “Mes tâches” pour qu’un utilisateur puisse retrouver facilement ses tâches créées
- Contrôle édition d’une tâche simultanée (simpliste)
  - Empêcher la soumissions d’un formulaire si la date de modification a changé entre temps
- Approvisionnement automatique des groupes de classes (étudier la faisabilité)
  - Préremplis avec certains professeurs comme membre
- Vue Semestrielle (voir Gantt)
- Une vue récap/statistique globale pour un groupe

#### 2.2.2.4 Won’t have

- Pas de gestion d’édition collaborative simultanée (web socket, style google docs)
- Pas de gestion des tâches transversales (sur plusieurs groupes)

### 3 Analyse

La section suivante décrit la partie d'analyse et de conception qui a été faite en amont pendant le dernier semestre de troisième année.

#### 3.1 Problématique

La problématique est divisée en deux parties, une partie concernant les professeurs et une autre les élèves.

##### 3.1.1 Pour les professeurs

**3.1.1.1 Problème 1** Placer au mieux les devoirs et CP pour lisser la charge et savoir si la charge de travail est correcte

**3.1.1.1.1 Solution** Elle repose sur deux propositions, dans un premier temps fournir une vue qui permette au mieux de placer un CP ou un devoir (tout en discutant avec les élèves). Dans un deuxième temps il sera possible pour un élève de réagir sur un devoir à l'aide de réactions qui permettront d'évaluer un devoir (trop long/trop complexe/etc...). Les professeurs pourront alors voir si un devoir a occasionné beaucoup de réactions et donc s'il était adapté ou non. Ces réactions pourront entraîner par la suite des discussions avec les élèves pour améliorer le devoir et à fortiori la participation des élèves.

**3.1.1.2 Problème 2** Les profs distribuent souvent des consignes de manière orale ou alors sur des supports spécifiques, avec les différents outils disponibles, rendre les informations sur toutes les différentes plateformes peut être éreintant

**3.1.1.2.1 Solution** Possibilité de déléguer cette tâche aux élèves (ex. le devoir peut être donné de manière orale et être rapporté par un des élèves), étant plus nombreux il est plus facile pour les élèves de centraliser les informations qui leur sont nécessaires pour un devoir plutôt que de laisser cette action à une seule et unique personne.

##### 3.1.2 Pour les élèves

**3.1.2.1 Problème** Les devoirs sont notés et éparpillés sur plusieurs supports (physique ou digital) parmi les élèves, on se retrouve souvent avec un élève qui détient une partie de l'information et non toute l'information. Les élèves ont donc du mal à visualiser tous les devoirs en cours donnés par les professeurs. Il est alors compliqué de prévoir sa charge de travail avec des informations incomplètes

**3.1.2.1.1 Solution** Apporter une vue centralisée dans laquelle il est facile de visualiser les tâches à faire, l'accès collaboratif permet de réunir les fragments d'informations détenus par chaque élève en un seul endroit afin d'obtenir une information complète.

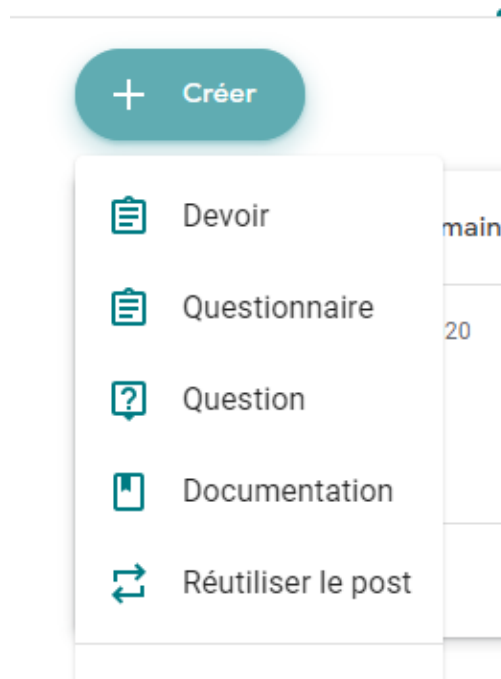
## 3.2 Etat de l'art

Cette section décrit l'actuel état des applications dans le domaine de gestion des tâches liés à des études, elle décrit notamment quelques tests effectués sur certaines de ces applications et ultimement le positionnement de Caravel par rapport à l'état de l'art.

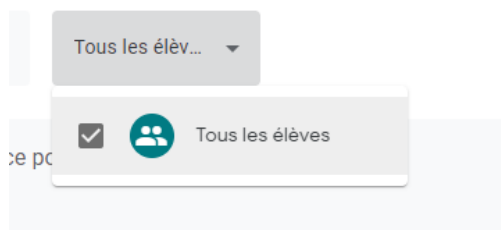
### 3.2.1 Applications testées

- Google Classroom (<https://classroom.google.com/>)
- MyHomework (<https://myhomeworkapp.com/home>)
- MyStudyLife (<https://app.mystudylife.com/>)

**3.2.1.1 Google Classroom** L'outil Google Classroom est très axé cours, les professeurs créent leur cours et les élèves suivent les cours qui leurs sont nécessaires (principe universitaire où les élèves peuvent choisir leur cours). L'idée est que les ajouts tels que les devoirs soient introduits uniquement par le professeur. La notion de classe à proprement parler n'existe pas.



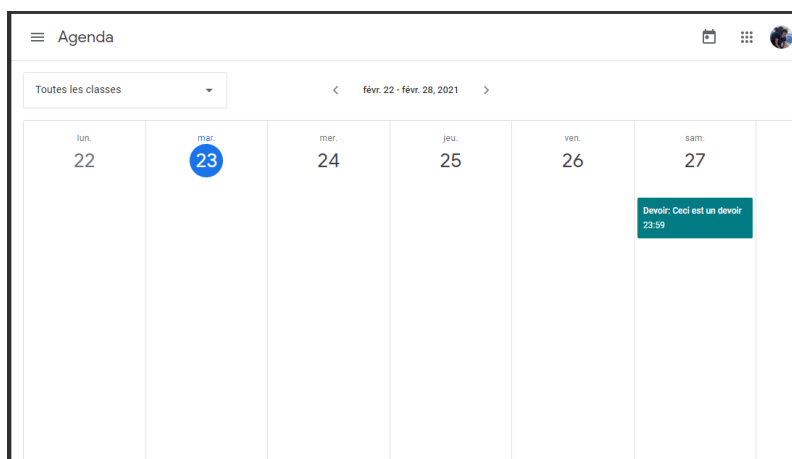
**Fig. 1 :** Google Classroom : Liste des types de devoirs



**Fig. 2 :** Google Classroom : Gestion de liste d'élèves



**Fig. 3 :** Google Classroom : Permet de faire des annonces



**Fig. 4 :** Google Classroom : Vue limité à la semaine

Fonctionnalités intéressantes :

- Des annonces peuvent être faites pour la classe

- Il y a une notion de groupes, les devoirs peuvent être distribué à toute la classe ou alors à un groupe plus restreint
- **Les devoirs sont synchronisés directement avec l'agenda Google**

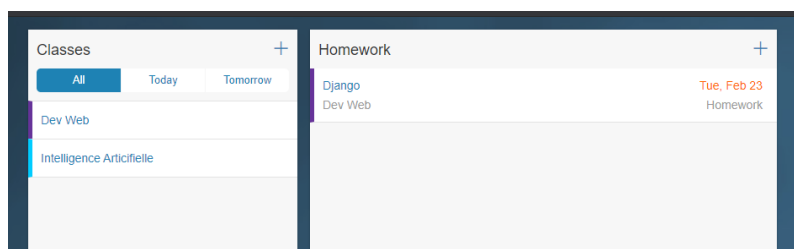
**3.2.1.2 MyHomework** MyHomework est une WebApp qui permet la gestion des devoirs personnels.

The screenshot shows a 'Homework' modal window with the title 'Add Homework' and a 'Close' button. The form contains the following fields:

- Description \***: Text input with 'TP3' entered.
- Class**: Dropdown menu with 'Intelligence Artificielle' selected.
- Type**: Dropdown menu.
- Due Date \***: Date picker showing '2/27/2021'.
- Due Time**: Text input.
- Repeats**: Dropdown menu.
- Priority**: Dropdown menu.
- Device Reminder**: Dropdown menu with 'Night before' selected.
- Additional Info**: Text area.
- ☐ **Complete**: Checkbox.
- File Attachments**: Text indicating 'Available on premium accounts, upgrade your account.'

At the bottom, there are three buttons: 'Save' (blue), 'Save & Add Another' (grey), and 'Cancel' (grey).

**Fig. 5 :** MyHomework : Ajout d'un devoir



**Fig. 6 :** MyHomework : Vue centrale simple et efficace

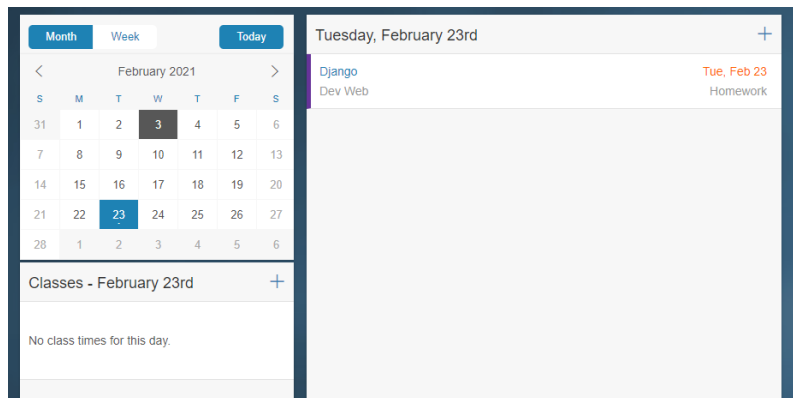


Fig. 7 : MyHomework : Vue calendrier (mensuelle)

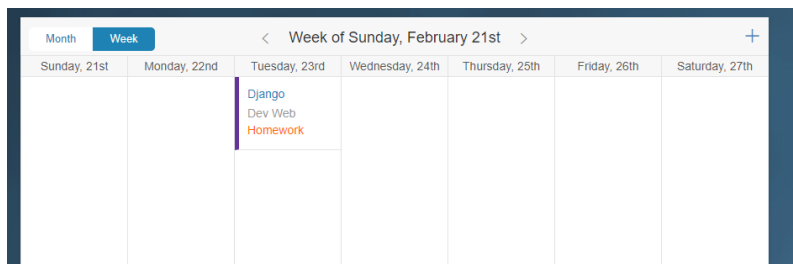


Fig. 8 : MyHomework : Vue semaine

L'application rend le service minimum (ajout de devoirs) mais ne permet pas la collaboration, il y a beaucoup d'options qui se relèvent pas très utiles dans la majorité des cas.

Fonctionnalités intéressantes :

- Affichage en semaine concise
- Possibilité de rajouter des rappels
- Ajout d'une notion "terminé"

### 3.2.2 MyStudyLife

MyStudyLife est une WebApp (disponible aussi sur mobile) qui permet la gestion des horaires de cours ainsi que des tâches à effectuer.

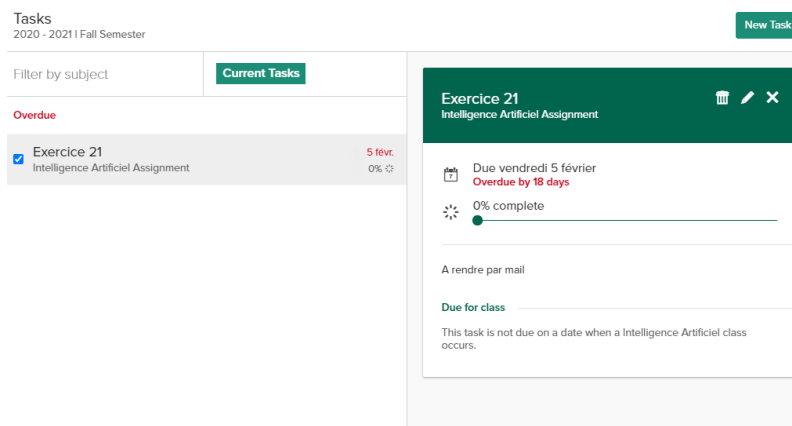


Fig. 9 : MyStudyLife : Gestion des tâches

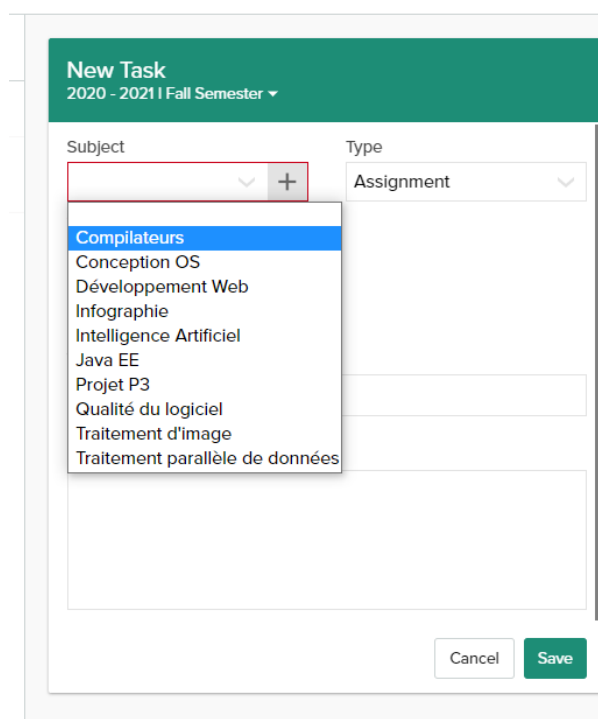


Fig. 10 : MyStudyLife : Gestion des différents sujets



Classes		Holidays	
0	Compilateurs: 3253.2 08:15 - 11:35 mercredi	FRT	Fall Break avr. 5 - 9 2021
1	Conception OS: 3253.1 10:00 - 11:35 lundi	CCO	Red Week mai 10 - 14 2021
	Conception OS: 3253.1 12:30 - 14:50 vendredi	CCO	
	Développement Web: 3255.2 13:15 - 14:50 lundi	DGR	
	Infographie: 3252.1 15:05 - 17:25 jeudi	BLC	
	Intelligence Artificiel: 3256.1 08:15 - 10:45 mercredi	STC	
	Intelligence Artificiel: 3256.1 08:15 - 09:45 jeudi	STC	
	Java EE: 3257.2 10:00 - 12:25 lundi	CHE_X	

Fig. 11 : MyStudyLife : Gestion des cours

Fonctionnalités intéressantes :

- Gestion des horaires de cours très flexible
- Permet de lier des tâches à un cours et d'effectuer des rappels avant le début du cours
- Permet de gérer les vacances
- Permet de gérer les cours sur différents semestres
- Les notifications push disponibles avec l'application sont très appréciable
- Possède une version web et mobile

### 3.2.3 Synthèse

Le marché est partagé en deux catégories : les applications de gestion de devoirs dans lesquelles c'est l'étudiant qui entre les devoirs et l'autre catégorie où ce sont les professeurs qui ajoutent les devoirs.

Pour la première catégorie, il existe actuellement beaucoup d'applications (surtout mobile) qui permettent à un étudiant de gérer ses devoirs mais celles-ci ne permettent pas la collaboration entre étudiants, en outre il n'est pas possible de partager les devoirs avec quelqu'un d'autre.

Pour la seconde catégorie, il existe quelques applications qui permettent à un professeur d'entrer des devoirs pour un groupe d'élèves, ceux-ci peuvent donc alors tous consulter les devoirs. Le problème repose sur le fait que seul le professeur peut entrer les devoirs, cela limite la marge de manœuvre des élèves ainsi que leur implication. Dans ces applications un professeur est souvent responsable de sa matière et ne peut donc pas forcément rajouter des devoirs dans une autre matière (voir Google Classroom).

### 3.2.4 Positionnement de Caravel

Dans Caravel l'idée est de se positionner entre les deux mondes, les devoirs seront gérées par les membres d'une classe, chaque membre de la classe a donc la possibilité de renseigner un devoir sur la plateforme. Si celui-ci manque de précision, le professeur ou un élève peut y apporter une modification en tout temps afin d'éclaircir son contenu. Dans cette optique autant les profs que les élèves peuvent être une source de devoir ce qui change la dynamique des élèves par rapport aux applications habituelles.

Un des souhaits de Caravel est aussi de réunir tous les devoirs en un seul endroit, avec l'apparition des différents outils comme Moodle, Teams, l'intranet ou encore les dossiers partagés, il est parfois très difficile de savoir où chercher l'information concernant un devoir. L'idée est donc de permettre aux membres de la classe de réunir les informations pertinentes en un seul endroit afin de gagner du temps.

De plus Caravel pourra permettre d'ouvrir des fils de discussion au sein d'une tâche afin de demander de l'aide ou des éclaircissement sur un détail spécifique, l'idée est de permettre aux élèves de s'entraider sur une question et de partager en un seul lieu les différentes informations sur un devoir, évidemment les professeurs peuvent aussi répondre aux différents fils de discussion.

Enfin, l'outil permet aux élèves de toujours être à jour quant aux tâches à faire, en effet il arrive souvent qu'un élève oublie de noter un devoir ou de détailler certains éléments, Caravel permet de réunir les connaissances des différents élèves et des professeurs pour obtenir une base solide d'informations.

### 3.3 Conception

Cette partie va s'atteler à décrire les problématiques ainsi que les solutions qui ont été trouvées.

#### 3.3.1 Rôles et use cases

Cette partie explicite les différents rôles disponibles au sein de caravel ainsi que les actions possibles.

**3.3.1.1 Rôles fonctionnels** Il y a un seul rôle fonctionnel qui donc celui d'administrateur du groupe, en général il s'agit du créateur du groupe mais ce droit peut être transmis.

**3.3.1.1.1 Administrateur du groupe** Permet de gérer les paramètres du groupe (suppression, renommage, etc...) ainsi que de gérer les différents membres du groupes (suppression d'un membre).

**3.3.1.1.2 Rôles sémantiques** Les rôles suivants n'auront pas de droits particuliers le but de l'application étant de permettre la collaboration directe entre les deux différents rôles, cependant les actions des professeurs seront mises en avant typiquement dans les fils de discussions. De plus les professeurs seront admis d'office dans les groupes dit de "classe" lors d'une demande d'adhésion.

Les deux rôles seront donc les suivants :

- Professeur
- Elève
  - Rôle par défaut

Il a été décidé de ne pas appliquer de droit spécifique pour l'un ou l'autre des rôles car un historique des actions de chaque utilisateur sera mis en place et donc il est possible en tout temps de trouver qui a effectué la moindre modification sur une tâche, il a donc été choisi de laisser libre tout utilisateur de modifier une tâche même s'il en est pas l'auteur. Ceci afin d'encourager la collaboration sur les différentes tâches.

### 3.3.2 Use cases

Pour la bonne compréhension des schémas qui vont suivre, il tenir compte du fait qu'un "Utilisateur" est un "Membre du groupe". De plus l'utilisateur est aussi considéré comme "auteur".

#### 3.3.2.1 Tâche Use cases concernant les différentes actions possibles sur les tâches.

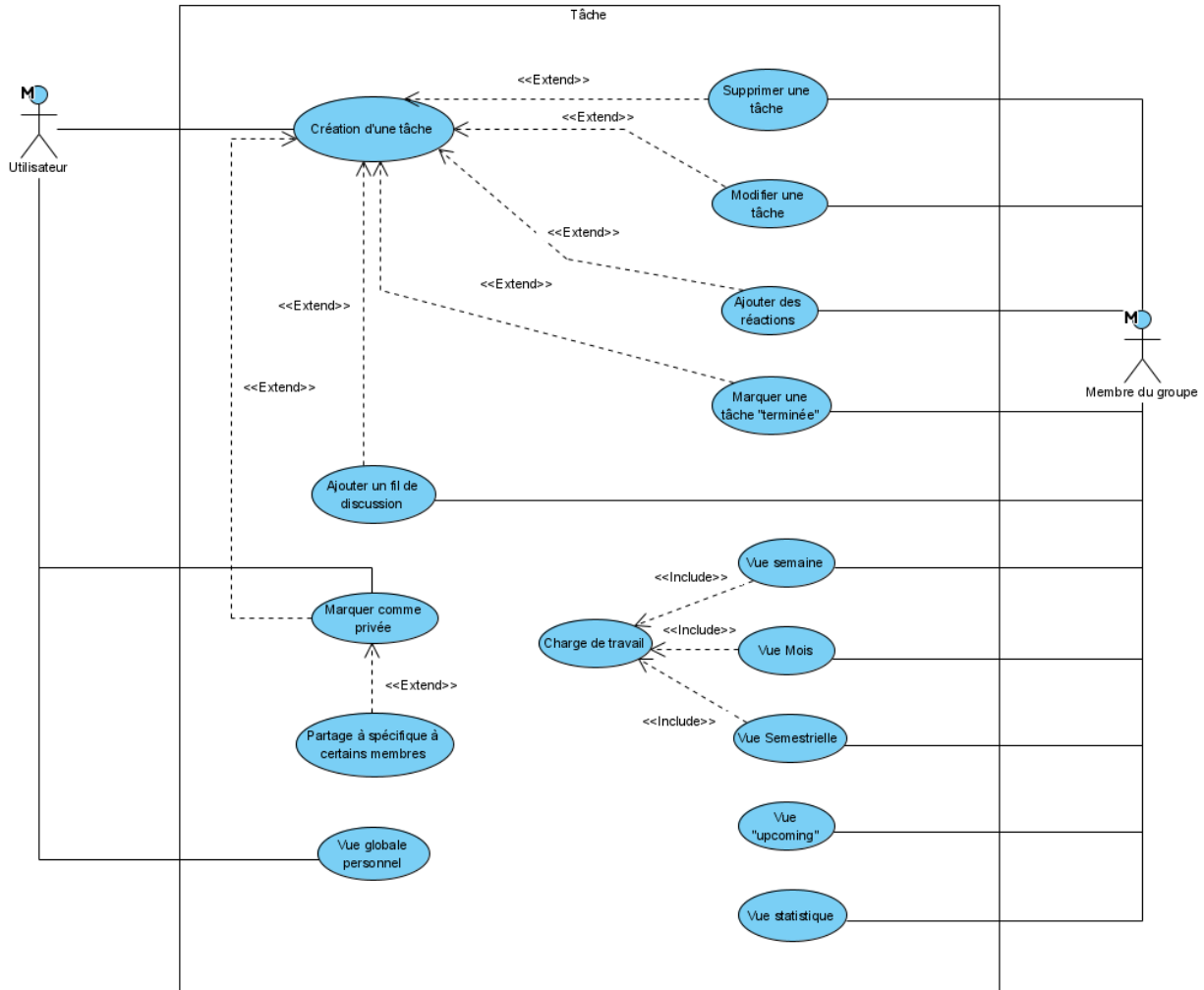
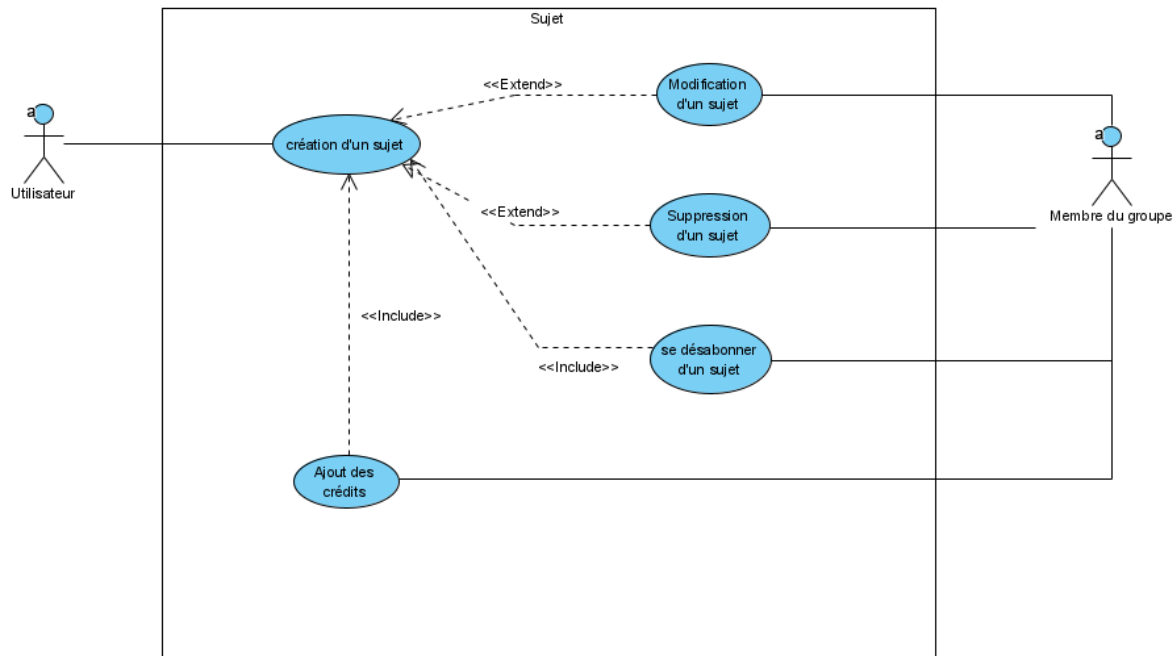


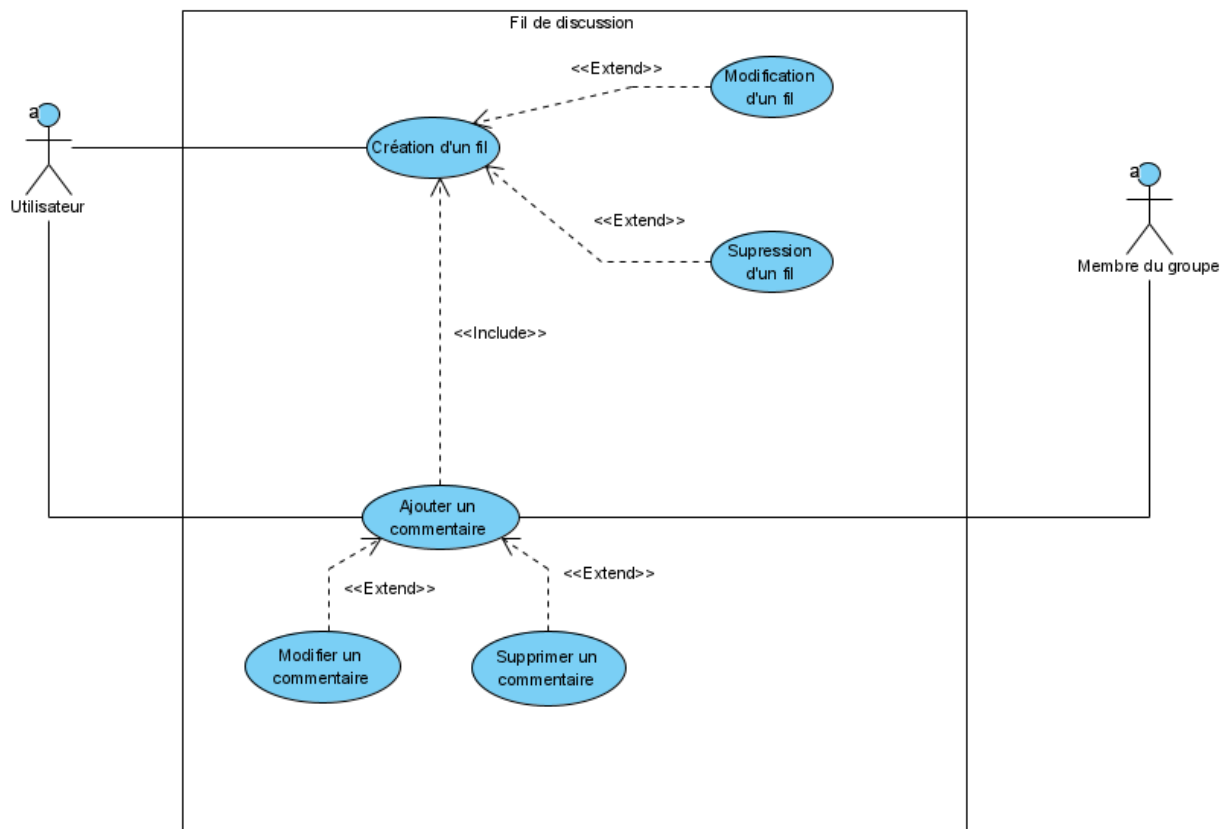
Fig. 12 : Use case : gestion des tâches

#### 3.3.2.2 Sujet Use cases concernant les différentes actions possibles sur les sujets.



**Fig. 13 :** Use case : gestion des sujets

**3.3.2.3 Fil de discussion** Use cases concernant les différentes actions possibles sur les fils de discussion.



**Fig. 14** : Use case : gestion des fils de discussion

**3.3.2.4 Notification** Les paramètres de notifications seront globaux et non spécifique à une tâche ou groupe

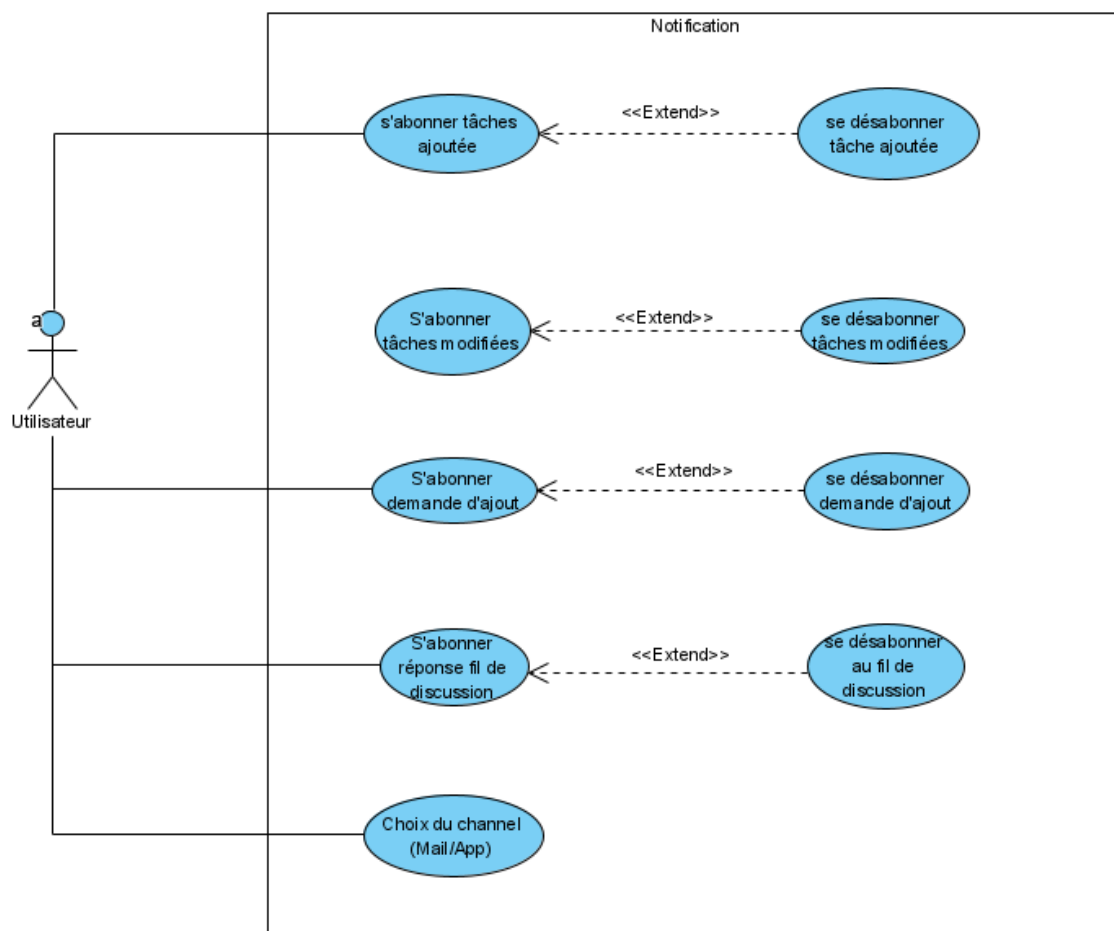
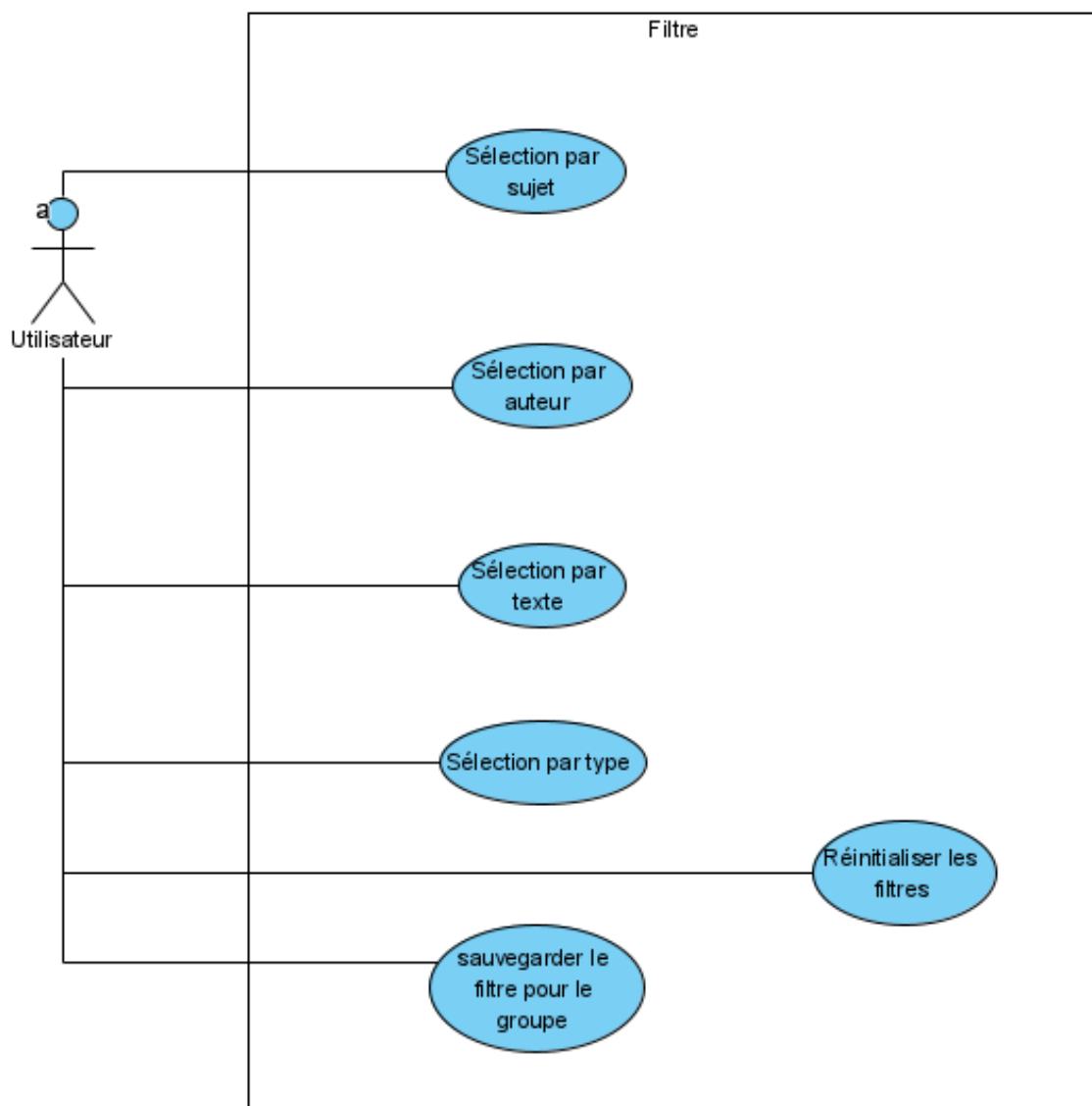


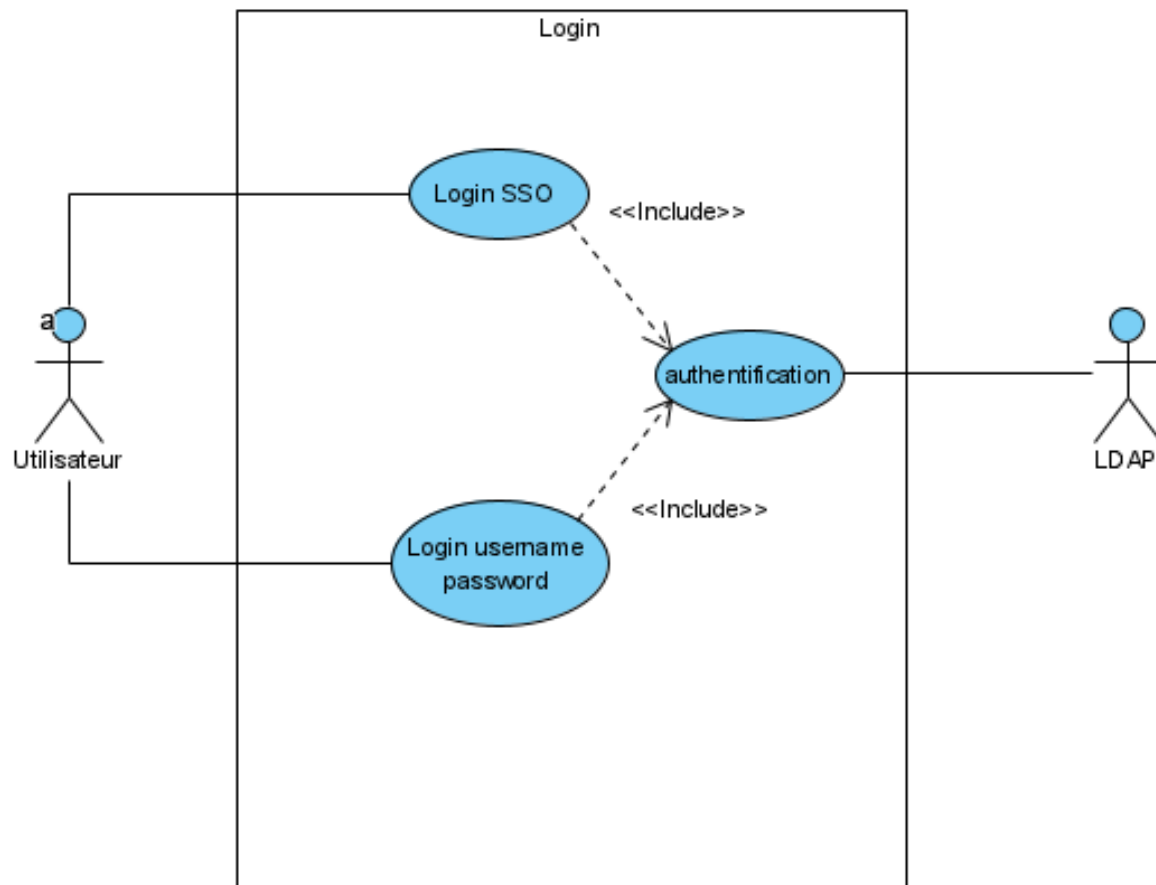
Fig. 15 : Use case : gestion des notifications

### 3.3.2.5 Filtre Use cases concernant les différentes actions possibles avec les filtres.



**Fig. 16 :** Use case : gestion des filtres

**3.3.2.6 Login** Use cases concernant les différentes actions possibles lors de l'authentification.



**Fig. 17** : Use case : gestion du login

*Les différents schémas ont été produits avec l'outil "Visual Paradigm community edition"*

### 3.3.3 Gestion des filtres

Les filtres permettent de filtrer la liste des tâches disponibles. Ils permettent de retrouver une tâche aisément et de retrouver une tâche passée ce qui n'est pas possible sur l'ancienne version de Caravel.

**3.3.3.1 Liste des filtres** Voici une liste exhaustive des filtres qui sont disponibles dans l'application :

- Par sujet
- Par auteur



- Par texte (texte entré par l'utilisateur, recherche dans le titre ainsi que la description)
- Par type de tâches (projet, devoir, CP/examen)
- Par état (clos/ouvert -> passé/futur)
- Par privé/publique







Il sera possible en tout temps de réinitialiser les filtres appliqués pour retrouver l'affichage standard.

### 3.3.4 Système de réactions

Le système de réactions sur les tâches a pour but de signaler la présence d'un problème sur cette dernière. L'idée derrière ces réactions est de donner une direction sur la réflexion à entreprendre pour évaluer la problématique d'une tâche dans cette optique les réactions ont pour but de juger de manière qualitative (en terme méthodologique) une tâche.

Pour ce faire nous allons définir une base de réactions qui sera la même sur toutes les tâches et que les élèves pourront utiliser. Il est important que cette base soit commune à toutes les tâches afin que les élèves puissent bien appréhender correctement leur utilisation. En effet des réactions spécifiques à chaque sujet ou groupe demanderait un effort d'assimilation trop conséquent et placerait l'élève dans une situation d'incertitude quant au choix de la réaction ce qui serait contre productif. Le système doit rester simple et pouvoir être assimiler facilement.

La liste exhaustives des réactions :

-  Trop long
-  Trop complexe
  - manque de compétences
-  Manque d'informations
  - donnée pas claire
-  Je suis perdu
  - la préparation en cours n'est pas optimale pour entreprendre l'exercice
-  Lien avec le cours pas clair
  - l'intérêt n'est pas clair, pas assez motivée, l'importance du devoir n'est pas comprise par l'étudiant
-  Peu d'intérêt
  - Par exemple pas de feedback, l'étudiant ne voit pas d'intérêt de s'investir

Plusieurs références ont été utilisées pour déterminer ces réactions :

- Un article de journal écrit dans le American Journal of Engineering Education (AJEE) Wenshu et al. (2018)
- Ainsi que deux autres articles web Dr. Davis Bowman (2016) et Dutta (2020-04-01)

**3.3.4.1 Participation aux réactions** Afin de pousser les utilisateurs à réagir sur les différentes tâches, un système de trigger pourra être mis en place, en substance, il s'agit de regarder les tâches terminées dans un certains laps de temps très court (1-2 jours) selon un taux de probabilité défini : notifier l'utilisateur afin qu'il réagisse à une tâche, la réaction n'est pas obligatoire.

**3.3.4.2 Anonymisation** La question s'est posée quant à l'anonymisation des résultats, une réflexion a été portée en ce sens : le fait d'anonymiser les résultats n'apporte pas de désavantage tandis que l'inverse peut freiner les utilisateurs à donner leur avis. Le choix s'est donc porter sur des réactions anonymes.

### 3.3.5 Système de notifications

Les notifications sont des éléments importants car ils permettent aux utilisateurs de toujours rester à jour par rapport au contenu de leurs groupes.

**3.3.5.1 Canaux de distribution** Les différents canaux de distribution visés, sont :

- PWA / interne à l'application
- Email

**3.3.5.2 Déclenchement des notifications** La liste exhaustive des actions qui peuvent déclencher une notification :

- Ajout d'une tâche
- Modification d'une tâche
- Suppression d'une tâche
- Ajout d'une question dans une tâche
- Ajout d'un commentaire si abonné ou si auteur
  - (par défaut si un utilisateur répond à une question ou s'il est auteur, il devient automatiquement abonné)
- Demande d'ajout au groupe
- Accepté dans un groupe
- Refusé d'un groupe

Les différentes notifications peuvent être paramétrables depuis le compte de l'utilisateur.

	Mail	PWA/Interne
Trigger 1	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Trigger 2	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Trigger 3	<input type="checkbox"/>	<input type="checkbox"/>
Trigger 4	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

**Fig. 18 :** Maquette des paramètres de notifications

### 3.3.6 Gestion de la charge de travail

Afin de mieux estimer la charge de travail chaque branche accueillera un nouveau paramètre, le nombre de crédits ETCS (un crédit représente une charge de travail d'environ 25 à 30 heures de travail). Ces crédits servent de critère

de pondération pour les différentes branches.

**3.3.6.1 Indicateurs** A partir de ces données une cotation **par semaine** est créée : le Work Load Score (WLS). Il s'agit d'un ratio basé sur un autre indicateur, le "Week Effort Score" (WES) et de sa médiane définie par les semaines déjà effectuées.

Les détails des calculs sont donnés par les formules suivantes :

```

1  $$N_A = Nombre\ de\ devoirs$$
2
3  $$N_E = Nombre\ de\ Examens\ ou\ CP$$
4
5  $$N_{PS} = Nombre\ de\ projet\ en\ cours\ (qui\ ne\ sont\ pas\ à\ rendre)$$
6
7  $$N_{PW} = Nombre\ de\ projet\ à\ rendre$$
8
9  $$C_S = Nombre\ de\ crédit\ pour\ le\ sujet\ (cours)$$
10
11 $$Week\ Effort\ Score\ (WES) = \sum_{subjects} C_s * (N_E + N_A + N_{PW} + 2 * N_{PS})$$
12
13 $$Work\ Load\ Score\ (WLS) = \frac{WES}{\widetilde{WES}}$$

```

**3.3.6.2 Comptabilisation des projets** Les projets sont calculés de manières différentes car on comptabilise un projet sur lequel on doit travailler mais qui n'est pas à rendre cette semaine et un projet qui est à rendre dans dans le courant de la semaine, ce qui engendre en général plus de travail.

**3.3.6.3 Normalisation** Le nombre de crédit n'est pas normalisé car tous les cours ne seront pas forcément présents en tout temps et donc il n'est pas possible d'avoir une normalisation linéaire si des sujets viennent s'ajouter au fur et à mesure (dans le cadre où ces informations ne sont pas calculées en temps réel)

**3.3.6.4 Gestion des extremums** Le score de certaines semaines risque de poser des problèmes, il faut donc éviter les extremums afin d'avoir une tendance qui soit plus cohérente. Pour éviter ça, la médiane des semaines est utilisée afin d'évaluer si une semaine est plus ou moins chargée

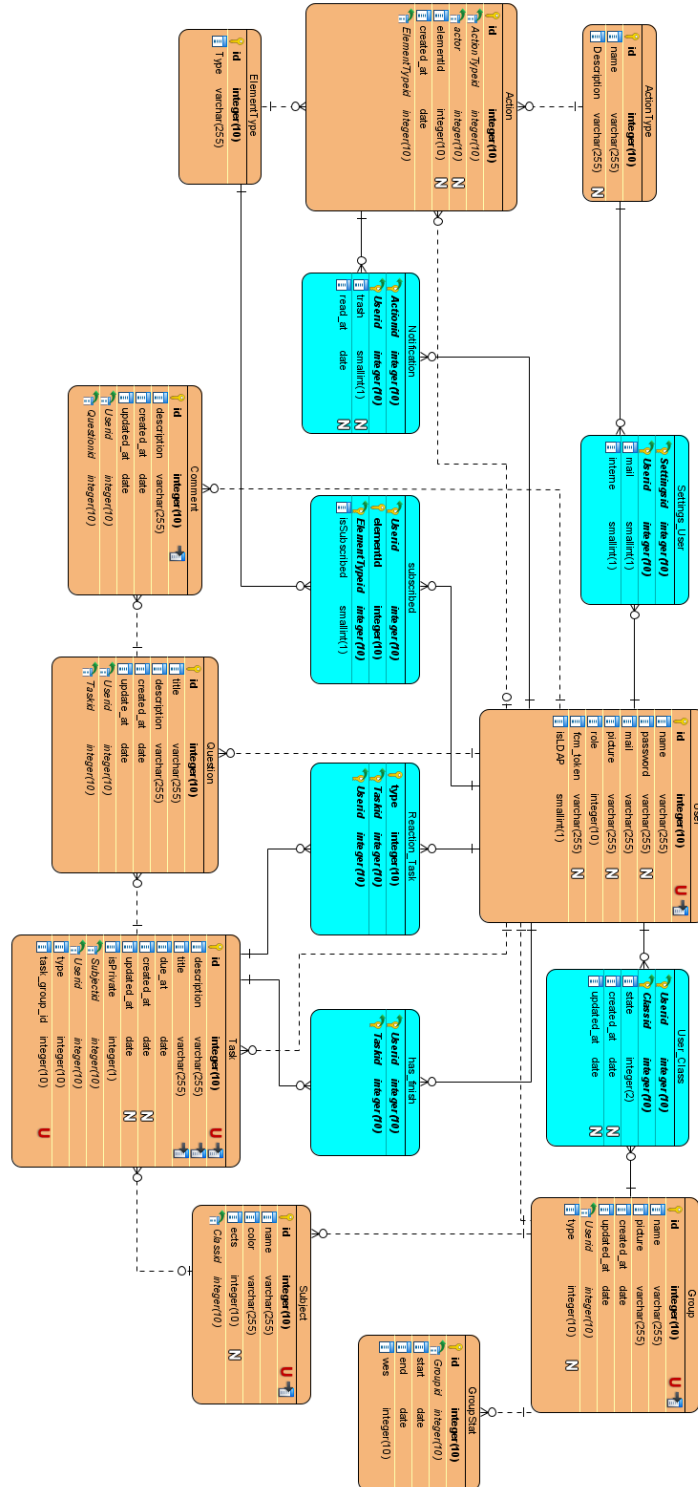
Projet N°227 : Caravel

He-Arc

21 juillet 2021

---

### 3.3.7 Modélisation de la base de données



### 3.3.8 Système d'authentification

L'actuel Caravel utilise un système de notification interne à l'application, il est donc nécessaire que chaque nouvel utilisateur d'enregistrer avant de pouvoir se connecter sur l'application. Afin de faciliter cette démarche, une solution de SSO avec Google et GitHub ont été mis en place. Dans la nouvelle version de Caravel, on souhaite permettre à l'utilisateur de se connecter via l'annuaire interne de l'école (LDAP). Cela permettra de directement récupérer des infos pertinentes sur l'utilisateur ainsi que de déterminer son statut (professeur ou élève). L'utilisation du LDAP devrait permettre à terme de pouvoir enrôler les utilisateurs directement dans des classes.

## 3.4 Définitions des routes

Les routes sont définies en utilisant le principe REST et donc avec l'utilisation des verbes HTTP : GET, POST, PUT/PATCH, DELETE. La génération des routes est documentée avec l'outil en ligne Swagger (Open API), sur lequel on peut retrouver la documentation de l'api Caravel<sup>1</sup>.

## 3.5 Stratégie & conception de test

Cette partie décrit la stratégie ainsi que la conception des tests nécessaire au bon fonctionnement de l'application. Il renseigne aussi les risques liés au projet.

### 3.5.1 Attentes de la qualité du produit

- Interface facile à utiliser
- Bonne qualité de code
- Site conforme aux normes standard du Web

### 3.5.2 Objectifs de tests

Le but des différents tests est de s'assurer que le code produit est de bonne qualité tant dans sa réalisation que dans son fonctionnement, en outre il permet de mettre en place des tests qui permettent de ne pas régresser d'une version à l'autre en maintenant une qualité de produit constante entre les différentes phases de développement.

- Avoir un code maintenable
- Avoir un bon temps de réponse
- Permettre une charge d'au moins 40 personnes

### 3.5.3 Périmètre de tests

- Test unitaire avec PHPUnit (et Jest côté Vue.js)
- Qualité de code avec SonarCloud

### 3.5.4 Gestion des risques

---

<sup>1</sup> <https://app.swaggerhub.com/apis-docs/M4n0x/Caravel/1.0.0#/>

Description	Source	Probabilité	Impact	Criticité	Résolution
Retard sur le planning	Interne	4	6	0.7	Découper les tâches de manière à facilement pouvoir évaluer le temps de mise en place (éviter les tâches avec trop d'action en même temps)
Login SSO	Interne	7	8	0.8	Voir la documentation, rapidement voir avec un professeur, réévaluer la faisabilité
L'appel des terrasses et de la bière	psychologique	10	5	0.5	boire de la bière sans alcool et éviter tout contact social, par exemple en effectuant du télétravail, afin d'éviter les collègues, ces tentateurs
Mauvaise évaluation de la charge de travail du à l'absence de connaissance approfondie sur certaines technologies	interne	8	7	0.9	En référé le plus rapidement possible au mandant et adapter les objectifs en fonction de retard pris

### 3.5.4.1 Etapes principales

1. Tests unitaires PHPUnit (et Jest côté VueJS)
2. Tests avec utilisateurs
3. Analyser la qualité de code avec SonarCloud
4. Analyser les résultats dans le rapport de tests

### 3.5.4.2 Environnement et outils de tests

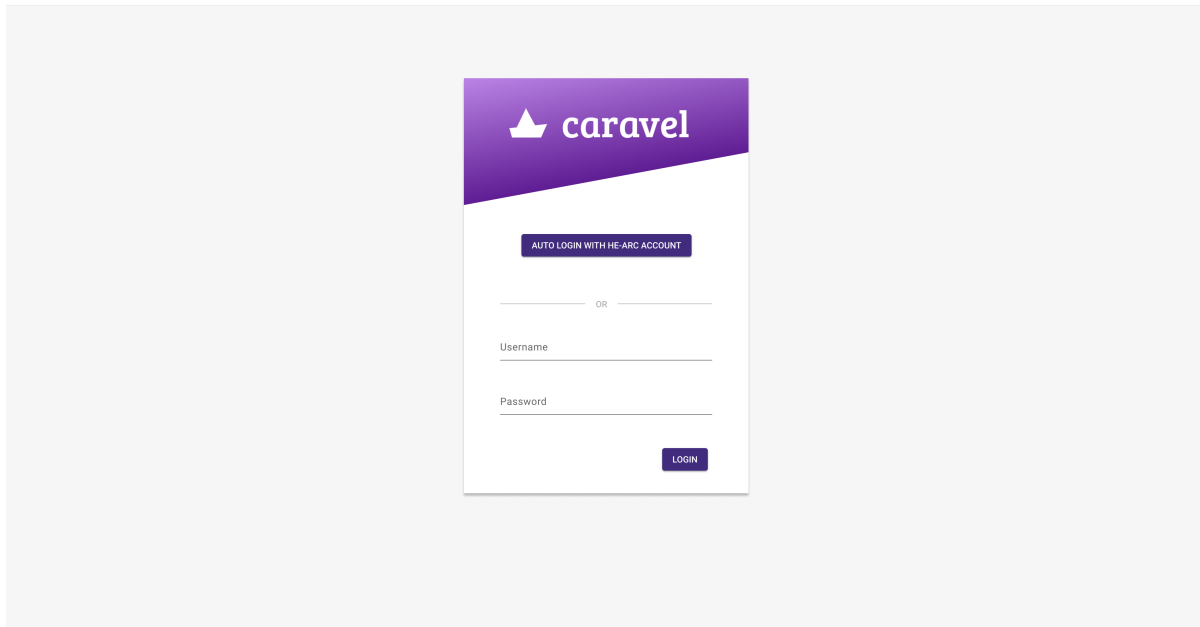
**3.5.4.2.1 GitHub** Pour l'intégration continue et la livraison continue GitHub sera utilisé.

**3.5.4.2.2 SonarCloud** La version cloud de SonarQube sera utilisée afin d'analyser la qualité du code.

## 3.6 Maquettes

Cette section regroupe les différentes maquettes créées pour la nouvelle version de Caravel. Ces maquettes ont été réalisées avec l'outil Figma avec une licence étudiante.

 caravel



**Fig. 20 :** Maquette : page de login



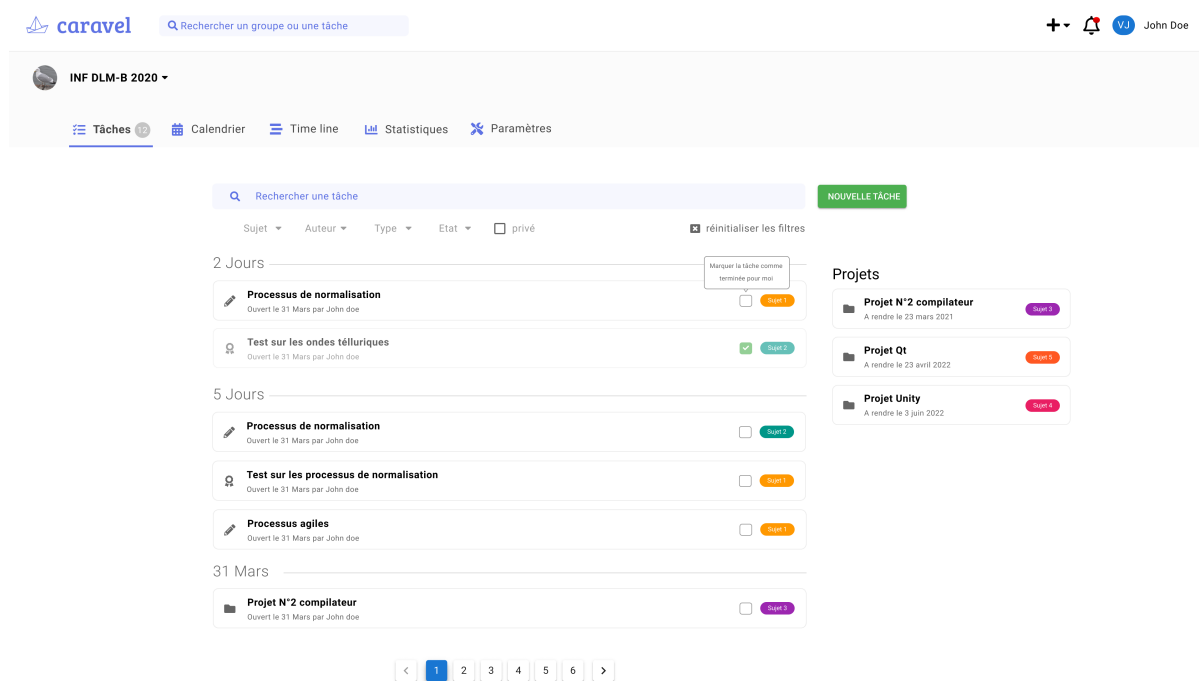


Fig. 21 : Maquette : page liste des tâches

caravel

Réchercher un groupe ou une tâche

+

John Doe

INF DLM-B 2020 •

TâchesCalendrierTime lineStatisticsParamètres

Ouvvert par John Doe le 31 mars · révisions •

SUIVREMARQUER COMME FAIT

Processus de normalisation

Jeudi, 19 août 2021 · 3 · 4

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis ut sagittis leo. Morbi quis blandit augue. Donec posuere aliquam mauris. Nunc auctor ligula non nisl congue pretium. Aliquam erat volutpat. Pellentesque in massa aliquet, suscipit est eu, consequat tellus. Curabitur sit amet euismod lorem, at mattis odio. Duis auctor sapien id risus feugiat sagittis. Cras facilisis ex sed velit commodo, non accumsan leo faucibus. Sed felis ipsum, malesuada nec iaculis eget, accumsan eu odio. Vestibulum sed nunc sit amet elit posuere pulvinar. Maecenas blandit imperdiet sodales. Ut aliquam justo et ante pharetra commodo. Nam faucibus, dui sit amet egestas eleifend, orci lectus porta orci, eget semper augue dui eget magna.

Donec laculus neque. Curabitur in velit mauris. Proin eleifend leo dapibus, feugiat sem vel, ornare nisi. Nulla vestibulum felis vel massa faucibus, sit amet convallis magna viverra. Pellentesque condimentum consectetur ullamcorper. Integer tempus nulla vitae nisl convallis, vel ornare mi placerat. Quisque eu magna arcu. Fusce rhoncus nec tellus in placerat. Sed vehicula, quam at rutrum tempus, nibh odio molestie lorem, id vulputate risus magna quis leo.

Nunc id ultricies neque. Curabitur in velit mauris. Proin eleifend leo dapibus, feugiat sem vel, ornare nisi. Nulla vestibulum felis vel massa faucibus, sit amet convallis magna viverra. Pellentesque condimentum consectetur ullamcorper. Integer tempus nulla vitae nisl convallis, vel ornare mi placerat. Quisque eu magna arcu. Fusce rhoncus nec tellus in placerat. Sed vehicula, quam at rutrum tempus, nibh odio molestie lorem, id vulputate risus magna quis leo.

2 questions

Comment résoudre l'exercice 1 sans trigonométrie ?

Képis

Jeudi, 18 mars 2021 · 3 · John Doe

Erreur sur l'exercice 2 point 4a ?

Domin

Jeudi, 15 avril 2021 · 1 · John Doe

Ajouter une question

Title

Be specific and imagine you're asking a question to another person  
e.g. Is there an R function for finding the index of an element in a vector?

Body

Include all the information someone would need to answer your question

B I G Q L M U Y P H A S C E

Links images Styling/Headers Lists Blockquotes Code HTML Tables More

code

\*\*bold\*\* \*italic\* >quote

AJOUTER

**Fig. 22 :** Maquette : page affichage d'une tâche

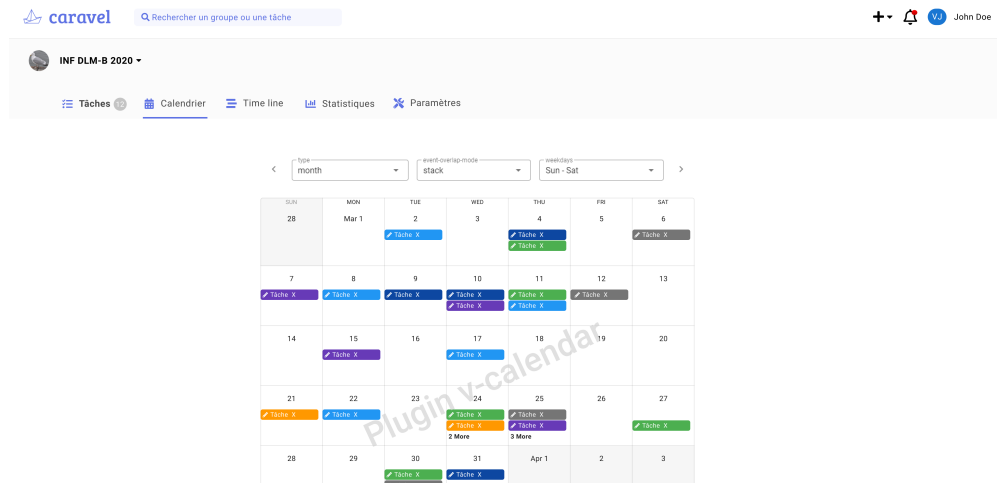


Fig. 23 : Maquette : page vue mensuelle

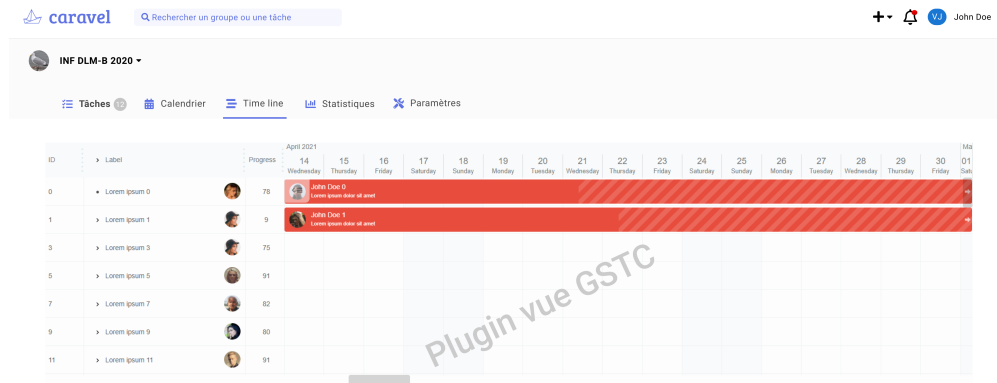


Fig. 24 : Maquette : page vue timeline

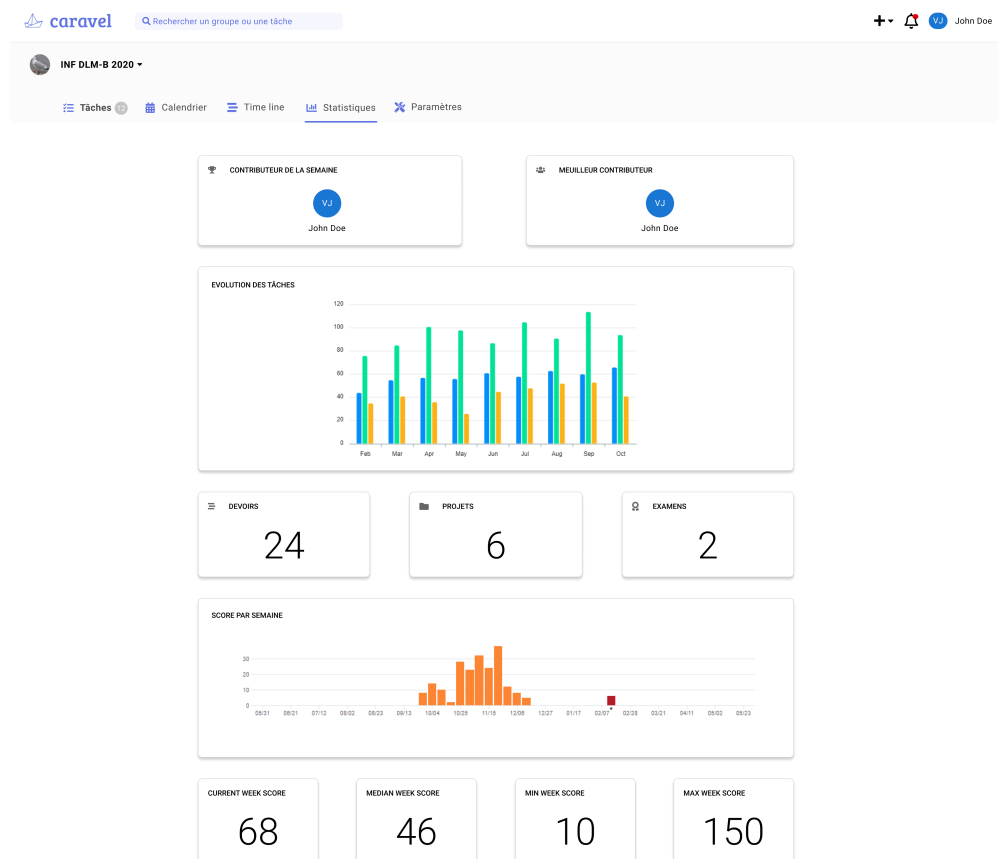


Fig. 25 : Maquette : page de statistiques

Il est de plus possible de consulter la version interactive<sup>2</sup> de la maquette directement sur le site de Figma.

### 3.7 Planning

voir annexe Planning.png

### 3.8 Méthodologie de travail

La méthodologie de travail se base sur l'utilisation GitFlow qui consiste à créer 3 différents canaux :

<sup>2</sup> <https://www.figma.com/proto/WHGPKvp8GgmoqsaOP7mFlz/Caravel-mockup>

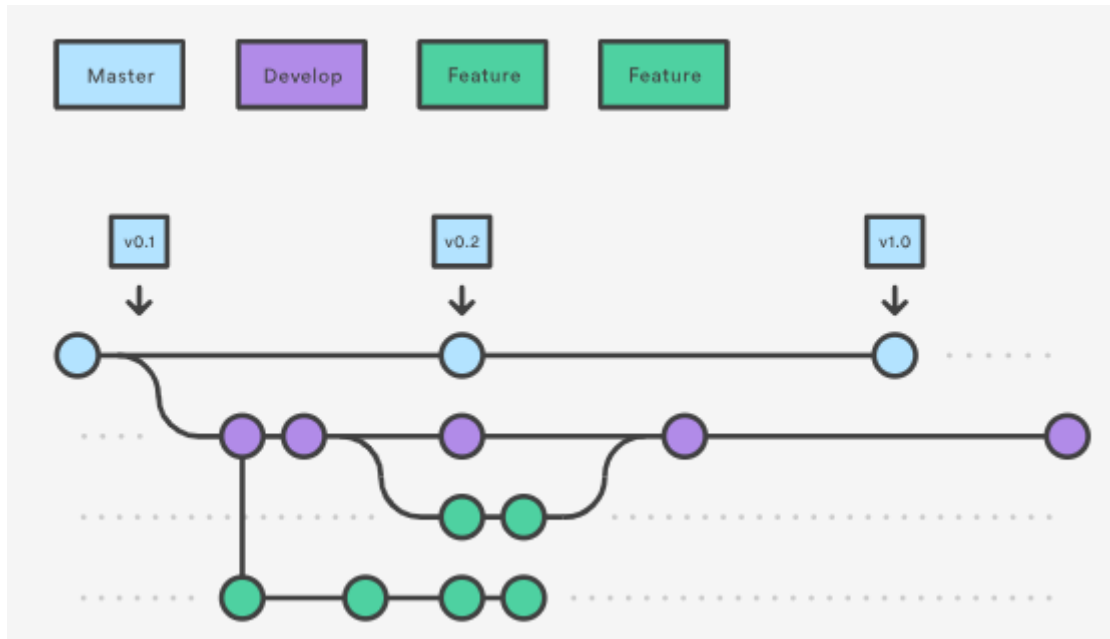


Fig. 26 : GitFlow workflow feature branches

1. La branche `master` qui est une branche qui est toujours fonctionnel et stable (release candidate)
2. La branche `develop` qui possède les dernières fonctionnalités mais n'est pas forcément stable
3. Les branches dites `features` qui sont créées pour chaque nouvelle fonctionnalité.

Lorsqu'une `feature` est aboutie et est validée par l'équipe de développement, elle est alors poussée sur la branche `develop` pour validation, pour une fois que la branche `develop` est considérée comme stable, celle-ci peut être poussée sur la branche `master`.

Cette méthodologie implique une bonne analyse en amont des tâches à effectuer ainsi qu'une découpage minutieux des tâches afin de garder des branches `features` simple et concise. Mais permet un suivi clair de l'avancement du projet ainsi qu'une revue plus simple de chaque nouvelle fonctionnalité mais demande un effort supplémentaire (création d'une branche et d'une pull request pour chaque fonctionnalité).

## 4 Implémentation

Dans cette section il s'agit d'expliquer les différentes étapes majeures qui ont permis la réalisation du projets ainsi que d'expliciter les différents choix techniques effectués.

### 4.1 Choix de la base de donnée

Pour le choix de la base de donnée, il y a globalement deux possibilités qui s'imposent : PostgreSQL ou MariaDB (MySQL). Un article de Kathuria (2019) compare ces deux versions en terme de performance, il s'avère que MariaDB est plus performant sur de grande requête que PostgreSQL. Le choix s'est donc porté sur l'utilisation de MariaDB.

### 4.2 Authentification

Le processus d'authentification un peu plus complexe dans une application où le frontend et le backend sont séparés, le processus peut être résumé simplement par le schéma suivant :

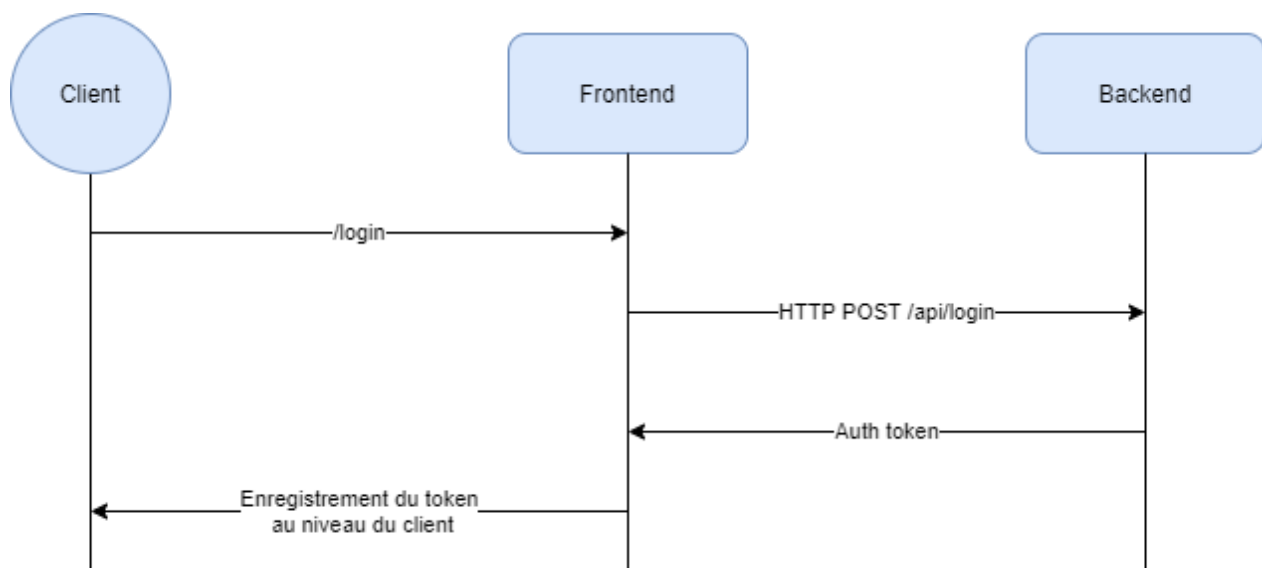


Fig. 27 : Schéma d'interaction pour l'authentification

#### 4.2.1 Local Storage vs Cookies

La complexité réside dans le choix du stockage du token au niveau du client, en effet une des solutions les plus utilisées est le stockage du token au niveau du Local Storage, cependant il s'agit d'une mauvaise pratique, comme le cite cet article de Degges (2018).

L'autre solution consiste à utiliser les cookies ainsi que le flag `"httpOnly"` qui bloque l'accès à ce dernier dès que ce flag est paramétré à vrai et c'est la solution qui est recommandée dans la documentation de Laravel, nous y

reviendront par la suite dans la section suivante.

#### 4.2.2 Sanctum vs Passport

Laravel propose deux systèmes d'authentifications, le premier Sanctum<sup>3</sup> est un système léger d'authentification basé sur des tokens, le second Passport<sup>4</sup> est un système d'authentification lourd qui utilise OAuth2, OAuth2 est un protocole qui permet aux utilisateurs la connexion avec d'autres applications externe tel que Google ou encore GitHub. Ce dernier est donc plus lourd et présuppose une bonne connaissance du protocole OAuth2. Comme l'utilisation de OAuth2 n'est pas nécessaire, Sanctum a été choisi, c'est d'ailleurs une recommandation issue de la documentation de Laravel<sup>5</sup>.

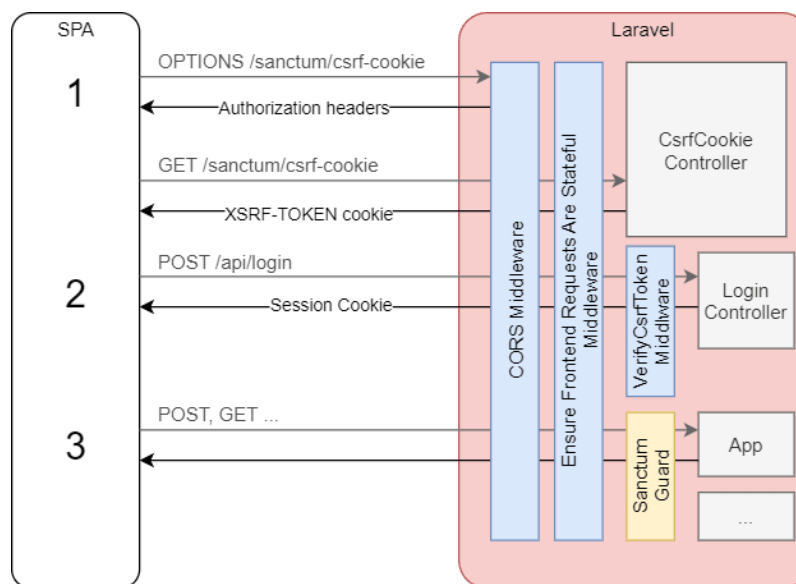


Fig. 28 : Laravel Sanctum Explained : SPA Authentication Bailly (2020)

Dans la figure 28 on peut voir le fonctionnement de sanctum, les éléments les plus importants sont le CSRF token ainsi que le Session cookie. Le CSRF token permet de protéger l'utilisateur d'une cross-site request cet élément n'est pas en httpOnly. Le Session Cookie, qui est le fonctionnement normal que nous pourrions retrouver avec une session PHP, est le cookie qui garde les informations liés à l'utilisateur, cet élément est protégé par un httpOnly et n'est donc pas accessible via javascript. Ces deux paramètres de session sont placés par Laravel à l'appel de la route `/sanctum/csrf-cookie`. Il est donc important de faire un appel à cette route avant toute tentative de connexion.

#### Code 1 : Fonction de login

<sup>3</sup> <https://laravel.com/docs/8.x/sanctum>

<sup>4</sup> <https://laravel.com/docs/8.x/passport>

<sup>5</sup> <https://laravel.com/docs/8.x/passport#passport-or-sanctum>

```
1  await axios.get(process.env.VUE_APP_API_BASE_URL + "sanctum/csrf-cookie");
2
3  const response: AxiosResponse = await axios({
4    url: process.env.VUE_APP_API_BASE_URL + "login",
5    data: { mail, password },
6    method: "POST",
7  });
```

Nous pouvons constater en ligne 1, aucun retour particulier n'est attendu car Laravel va automatiquement inscrire les cookies nécessaires et la librairie utilisée pour les appels backend, Axios, va lui aussi de manière automatique faire les configurations nécessaires dès lors que le paramètre `axios.defaults.withCredentials = true`; est positionné.

#### 4.2.3 LDAP

TODO

### 4.3 DevOps CI/CD

Cette section explique comment a été mise en place le déploiement automatique ainsi que les différentes pipeline de test. Les détails d'implémentation de configuration des différentes pip

#### 4.3.1 Intégration continue

L'intégration continue consiste à faire des livraisons continue ainsi que de mettre en place des tests afin de vérifier que ces livraisons soient stables. Ces livraisons continue sont en partie réalisé grâce à la méthodologie GitFlow et aux GitHub actions

**4.3.1.1 Laravel** Laravel possède deux pipelines de test, une qui utilise MySQL (MariaDB) et une autre SQLite, cette façon de faire nous garanti une certaine abstraction entre l'utilisation de la base de donnée et notre code, en effet chaque moteur de base de données possède des particularités en utilisant deux systèmes de base de données on peut garantir l'interpolation entre les deux différents type de base de donnée.

En plus du test de connexion à la base de donnée, les tests unitaires PHPUnit sont lancés en fin de traitement pour les deux pipelines.

#### Code 2 : Pipeline de test Laravel SQLite

```
1  name: Laravel CI SQLite fast
2
3  on:
4    push:
5
6  defaults:
7    run:
8      working-directory: ./backend
9
10 jobs:
11   tests:
```



```

12     runs-on: ubuntu-latest
13
14     steps:
15     - uses: actions/checkout@v2
16     - name: Copy .env
17       run: php -r "file_exists('.env') || copy('.env.example', '.env');"
18     - name: Setup PHP
19       uses: shivammathur/setup-php@v2
20       with:
21         php-version: '7.4'
22     - name: Install composer Dependencies
23       run: composer install -q --no-ansi --no-interaction --no-scripts --no-progress --prefer-dist
24     - name: Generate key
25       run: php artisan key:generate
26     - name: Directory Permissions
27       run: chmod -R 777 storage bootstrap/cache
28     - name: Create Database
29       run: |
30         mkdir -p database
31         touch database/database.sqlite
32     - name: Execute tests (Unit and Feature tests) via PHPUnit
33       env:
34         DB_CONNECTION: sqlite
35         DB_DATABASE: database/database.sqlite
36       run: |
37         php artisan migrate --seed
38         vendor/bin/phpunit

```

La version MySQL est similaire à cette dernière, seule la configuration de la base de donnée change.

#### 4.3.2 Livraison continue

La livraison est une étape qui consiste à déployer de manière automatique dès qu'une modification de code est effectuée. Ainsi notre application reflète toujours l'état actuel du code.

**4.3.2.1 Configuration du serveur** Le déploiement automatique ne s'occupe que de mettre les données de l'application à jour, elle ne s'occupera pas de la configuration totale du serveur qui nécessite plusieurs composants indépendants (npm, nginx, composer, php, mariadb, etc...). Il faut donc s'atteler à créer une configuration minimale du serveur pour accueillir notre backend ainsi que notre frontend.

```
""{.bash caption="Serveur : installation des dépendances de base} #Server database mariadb sudo apt install mariadb-server
```

```
# NPM pour VueJs sudo apt install npm
```

```
# PHP et de ses dépendances sudo apt install php7.4 libapache2-mod-php7.4 php7.4-curl php-pear php7.4-gd php7.4-dev php7.4-zip php7.4-mbstring php7.4-mysql php7.4-xml curl php7.4-ldap -y
```

```
# Composer pour les dépendances php sudo apt install composer
```

```

1  ##### Configuration de MariaDB
2
3  Enfin de garder la base de donnée de Caravel dans un vase clos, un utilisateur spé
    cifique est créé pour accéder aux données de Caravel.
4
5  ```{.sql caption="Serveur : création de la db et d'un user particulier"}
6  CREATE DATABASE Caravel;
7
8  grant all privileges on Caravel.* TO 'Caravel'@'localhost' identified by '
    PLACERHOLDER_PASSWORD';
9
10 flush privileges;
```

**4.3.2.2 Configuration de Nginx** Dans un premier temps, il a été choisi de séparer le backend et le frontend via deux sous domaines différents à savoir <http://caravel.ing.he-arc.ch/> pour la partie frontend et <http://api.caravel.ing.he-arc.ch/> mais après plusieurs tests, il semble que cela ne soit pas possible en tout cas en l'état, car le sous domaine `api.*` n'est pas redirigé sur le serveur, il faudrait donc ajouter une entrée DNS supplémentaire au niveau du service informatique. Afin de ne pas perdre de temps sur la partie configuration, il a été choisi en définitive de faire passer l'api sur une route spécifique c'est-à-dire <http://caravel.ing.he-arc.ch/api>. La configuration suivante du Nginx reflète ce dernier choix.

**Code 3 :** Serveur : configuration Nginx

```

1  # Caravel conf
2  server {
3      index index.php index.html;
4      root /var/www/caravel/backend/public;
5      server_name caravel.ing.he-arc.ch;
6      client_max_body_size 210M;
7
8      location / {
9          root /var/www/caravel/frontend/dist;
10         try_files $uri /index.html;
11     }
12
13     location /api {
14         try_files $uri $uri/ /index.php?$query_string;
15     }
16
17     location ~ /uploads/ {
18         root /var/www/caravel/backend/public/;
19     }
20
21     location ~ /\.php$ {
22         fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
23         fastcgi_index index.php;
24         fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
25         include fastcgi_params;
26     }
```

```

27
28     listen 443 ssl http2; # managed by Certbot
29     ssl_certificate /etc/letsencrypt/live/caravel.ing.he-arc.ch/fullchain.pem; #
        managed by Certbot
30     ssl_certificate_key /etc/letsencrypt/live/caravel.ing.he-arc.ch/privkey.pem; #
        managed by Certbot
31     include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
32     ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
33 }
34
35 server {
36     if ($host = caravel.ing.he-arc.ch) {
37         return 301 https://$host$request_uri;
38     } # managed by Certbot
39
40
41     listen 80;
42     server_name caravel.ing.he-arc.ch;
43     return 404; # managed by Certbot
44 }
```

La partie SSL est directement gérée par CertBot<sup>6</sup> qui est un outil open source qui permet d'automatiquement enrôler les sites issue de la configuration nginx avec des certificats Let's Encrypt et donc d'activer l'HTTPS.

**4.3.2.3 Runners GitHub** Un runner GitHub est un agent qui s'installe sur un serveur distant, il permet de lancer directement des tâches sur ce dernier depuis des GitHub Actions<sup>7</sup>, cela est nécessaire pour effectuer le déploiement automatique. Pour les détails d'installation voir la page wiki dédiée<sup>8</sup> sur le GitHub de Caravel.

### 4.3.3 Environnement de production

Certaines configurations sont dépendantes de l'environnement (base de données, ldap, etc...), un fichier de configuration unique ne peut être défini pour tous les environnements. De même que ces derniers fichiers peuvent contenir des informations sensibles il est donc nécessaire de faire une configuration propre à chaque environnement dont elle seul détient les informations.

**4.3.3.1 Méthode par écrasement** La méthode pour gérer la configuration de la production consiste à écraser ou ajouter les fichiers nécessaires lors du déploiement automatique, pour ce faire la configuration est stockée sur un dossier spécifique `/var/www/config/caravel` et à chaque déploiement cette configuration est copiée dans le répertoire de déploiement. Il suffit donc de poser le fichier de configuration nécessaire que l'on veut surcharger dans le dossier en respectant la nomenclature du dossier cible, par exemple pour le fichier `auth.php` il faut donc déposer le fichier `auth.php` dans `/var/www/config/caravel/backend/config/` il sera alors automatiquement copié lors du déploiement.

<sup>6</sup> <https://certbot.eff.org/>

<sup>7</sup> <https://github.com/features/actions>

<sup>8</sup> <https://github.com/HE-Arc/Caravel/wiki/CI-CD-Caravel-2.0>

**4.3.3.2 Dossier de téléchargement** Une des problématiques avec le déploiement c'est que la mise à jour passe par le remplacement de tous les fichiers de notre application par la dernière version disponible sur notre repository GitHub, cependant les dossiers de téléchargement sont propres à chaque webapp, il ne faut donc pas les remplacer lors de la mise à jour du contenu de notre webapp.

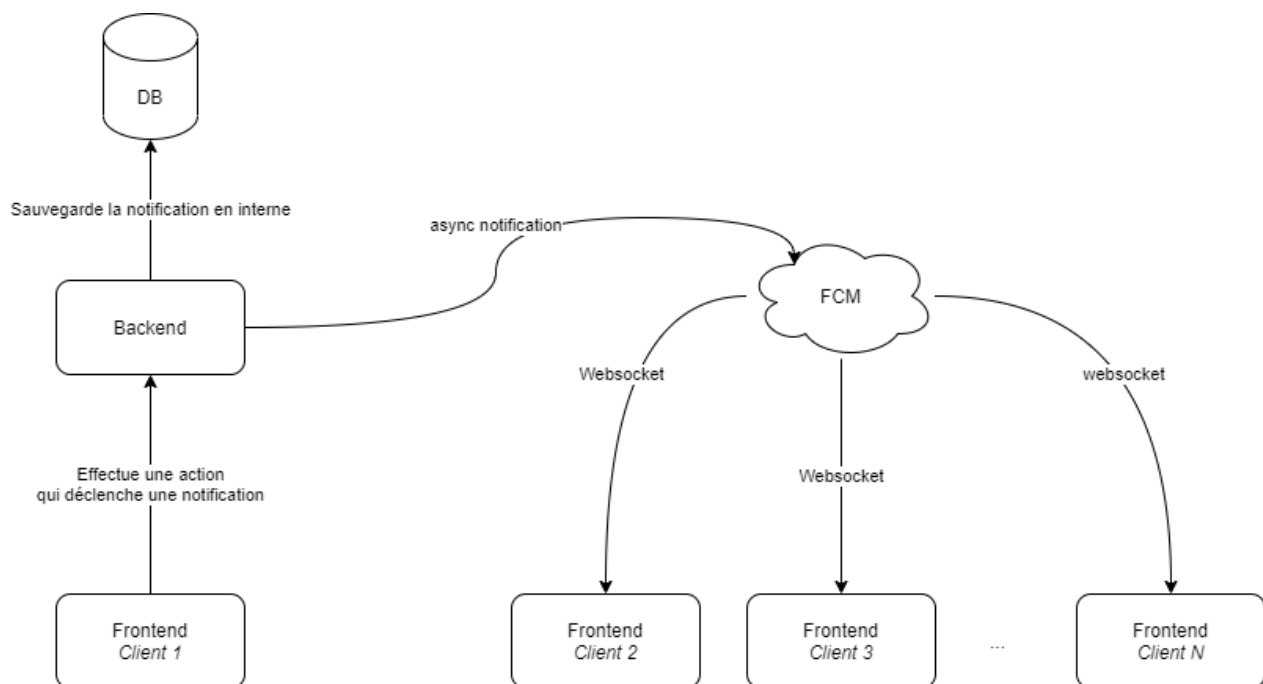
Pour régler cette problématique, les deux dossiers de téléchargement :

- `backend/storage`
- `backend/public/uploads`

sont, lors du déploiement, créés comme des liens symboliques sur des répertoires qui ne sont pas dans le répertoire de déploiement, ce qui évite que les dossiers ne soient à chaque fois écrasés pour le déploiement d'une nouvelle version.

## 4.4 Système de notification

Le système de notification se divise en deux parties, la première qui est le déclencheur des notifications et la seconde qui consiste à récupérer les notifications. Ce découpage en deux parties est relative au découpage back et front end.



**Fig. 29** : Schéma global du système de notifications

Sur la figure 29 nous pouvons voir le schéma global des transactions effectuées lors d'une notification. Le déclenchement qui se produit avec le client 1, puis au niveau du backend nous avons deux actions qui sont effectuées,

une première va enregistrer la notification en DB, l'autre va s'occuper d'envoyer une notification au serveur Firebase Cloud Messaging. Et finalement les différentes notifications vont être descendues sur les différents clients via un système de websocket mis en place grâce aux outils fournis par Firebase.

#### 4.4.1 Déclenchement d'une notification

Pour déclencher une notification au niveau du backend, il faut qu'une modification ait lieu sur une tâche, une question ou un commentaire. Pour détecter ses changements des observateurs<sup>9</sup> ont été mis en place au niveau du backend. Dès qu'une action parmi les types Création, Mise à jour et Suppression est effectuée, ces observateurs sont susceptibles d'être appelés. Chaque observateur a sa propre mécanique pour savoir quand il doit être déclenché et à qui les notifications sont destinées.

**4.4.1.1 TaskObserver** Le TaskObserver est déclenché par les actions suivantes :

- Création
- Mise à jour
- Suppression

Lorsqu'une de ces actions est effectuée, le TaskObserver est alors appelé et les notifications sont envoyées directement à tous les membres du groupe auquel la tâche est rattachée.

**4.4.1.2 QuestionObserver** Le QuestionObserver est déclenché par les actions suivantes :

- Création
- Mise à jour

Lorsqu'une de ces actions est effectuée, le QuestionObserver est alors appelé et les notifications sont envoyées directement à tous les membres du groupe auquel la question est rattachée.

**4.4.1.3 CommentObserver** Le CommentObserver est déclenché par les actions suivantes :

- Création

Lorsqu'une de ces actions est effectuée, le CommentObserver est alors appelé et les notifications sont envoyées directement à tous les participants de la question.

**4.4.1.4 Gestion de la diffusion des notifications** Lorsqu'une notification est créée, elle représente une classe particulière, la classe `Action.php`, c'est elle qui va enduire le comportement de la notification, c'est à dire comment elle va être distribuée ou stockée. Elle possède donc une méthode propre qui donne les différents canaux de diffusion de la notification.

#### Code 4 : Notifications : canaux de diffusion

```
1 /**
2  * Get the notification's delivery channels.
```

<sup>9</sup> <https://laravel.com/docs/8.x/eloquent#observers>

```

3      *
4      * @param mixed $notifiable
5      * @return array
6      */
7      public function via($notifiable)
8      {
9          //faire le check pour le user (notifiable) si les paramètres sont ok
10         return ['database', FcmChannel::class];
11     }

```

Dans notre cas, notre action va être stockée en DB et diffusée via le Firebase Cloud Messaging. Pour le canal `database` il s'agit d'un canal disponible par défaut dans Laravel, en ce qui concerne le FCM channel il s'agit d'un ajout externe<sup>10</sup>.

Pour chaque canal il faut ensuite déterminer son comportement via les méthodes appropriées.

#### Code 5 : Notification : comportement des canaux

```

1      public function toFcm($notifiable) // Pour Firebase Cloud Messaging
2      {
3          return ...;
4      }
5
6      public function toArray($notifiable) // Pour l'enregistrement en DB
7      {
8          return ...;
9      }

```

**4.4.1.4.1 Envoi asynchrone des notifications** Si beaucoup de membres sont dans le groupe cette action peut prendre beaucoup de temps, il est nécessaire que cette tâche ne bloque pas la requête du client, il est possible de faire en sorte de mettre les notifications dans une queue qui sera alors exécuté dans un autre thread. Pour cela il suffit de rajouter le trait `Queueable` à notre classe ainsi que l'interface `ShouldQueue`

#### Code 6 : Notification : envoi asynchrone

```

1      class ... implements ShouldQueue
2      {
3          use Queueable;
4
5          ...
6
7      }

```

A partir de là Laravel s'occupe seul de faire le travail en détectant automatiquement l'interface `ShouldQueue`.

<sup>10</sup> <https://laravel-notification-channels.com/fcm/>

#### 4.4.2 Configuration de FCM

Pour la configuration de FCM au niveau du backend la documentation officiel de El-Din (2020-03-04) doit être suivie. En ce qui concerne la configuration au niveau du backend il faut se référer aux documents utilisés dans le cadre de ce projet :

- How to add FCM to vue.js, Taf (2021-03-08).
- Intégration de Firebase Cloud Message avec Laravel et Vue.js, Christiandy (2019).
- Documentation officiel de Firebase Cloud Message, Google (2021).

**4.4.2.1 Changements par rapport à la conception** Par rapport à la conception les déclencheurs suivants n'ont pas été traités :

- Demande d'ajout au groupe
- Accepté dans un groupe
- Refusé d'un groupe

par manque de temps.

#### 4.4.3 Récupération des notifications depuis le frontend

Pour l'envoi de notification aux utilisateurs, le backend a besoin de connaître le token FCM de l'utilisateur, ce token ne peut être obtenu que par le client, comme la notification est lancée depuis le backend pour des raisons de sécurité il faut donc transmettre ce token du front au backend.

Cette transaction se fait depuis la page de login, si l'utilisateur accepte les notifications push, alors l'enrollement du token FCM est effectué, une route au niveau de l'api `api/profile/fcmToken` est donc disponible pour enregistrer le token depuis le frontend.

### 4.5 Frontend

`typescript`

#### 4.5.1 Vuex

#### 4.5.2 Localisation

#### 4.5.3 Composants

##### 4.5.3.1 Pagination

##### 4.5.3.2 Inputs

#### 4.5.4 Filtres

#### 4.5.5 Gestion des erreurs Axios

#### 4.5.6 Gestion du chargement

### 4.6 Backend

#### 4.6.1 Politiques

#### 4.6.2 Validation des requêtes

FormRequest

#### 4.6.3 Localisation

#### 4.6.4 Moteur de recherche

#### 4.6.5 Gestion de clés locales

## 5 User tests

Afin de tester globalement l'application la réalisation d'un test utilisateur à eu lieu à St-Imier, le 07 juillet 2021.

Il y a eu au total 6 personnes interrogées dans le cadre de ce user test, les résultats des différents retours par les utilisateurs sont décrits dans le paragraphe qui suit, il s'agit essentiellement de données brutes.

### 5.1 Scénario

- Connectez-vous sur Caravel
- Vous êtes dans la classe "INF DLM-B 2021" et vous souhaiteriez créer un espace pour votre classe sur Caravel
- Vous vous souvenez d'un devoir pour le mardi 20 juillet pour le cours d'infographie : "Lire tout le livre de WebGL" que vous souhaiteriez partager à votre classe à travers Caravel.
- Vous prenez connaissance de l'existence du groupe "INF DLM-B 2019" et vous décidez de rejoindre le groupe.
- Un utilisateur vient d'accepter votre demande d'accès à la classe "INF DLM-B 2019", vous souhaitez maintenant accéder au groupe pour voir ce qu'il contient.
- En arrivant sur le groupe "INF DLM-B 2019" vous apercevez la tâche "Faire l'exercice 1" d'infographie, vous prenez le temps de le lire et décidez que ce travail n'est pas de votre niveau, vous souhaitez réagir à la tâche pour montrer votre opinion.
- Malgré votre réaction, vous entreprenez quand même de réaliser le devoir, vous bloquez immédiatement sur le point 1a, vous souhaitez demander "Comment résoudre l'exo 1a" à vos camarades qui se trouvent sur le groupe.
- En parcourant les diverses tâches du groupe "INF DLM-B 2019" vous apercevez la tâche "Séance de travail", sur celle-ci se trouve une question "A quelle heure à lieu la séance?" Vous connaissez la réponse (17h30) et



décidez d'y répondre.

- En revenant sur votre question que vous avez posée "Comment résoudre l'exo 1a" vous apercevez que quelqu'un a répondu à votre question, la réponse vous convient, vous décidez que cette réponse est suffisante et passez l'état de la question en résolu.
- Grâce à l'aide fournie par vos camarades sur la tâche "Faire l'exercice 1" d'infographie, vous avez réussi l'exercice, vous décidez de marquer cette exercice comme terminé pour vous.
- Vous réalisez que la tâche que vous avez ajoutée "Lire tout le livre de WebGL" était une erreur vous décidez de supprimer cette tâche.
- D'ailleurs vous décidez que le groupe "INF DLM-B 2019" est beaucoup mieux que le groupe que vous avez créé, comme ce dernier n'est plus utile et qu'il n'y a que vous, vous décidez alors de supprimer le groupe.
- Finalement vous avez décidé de changer de classe, vous préférez donc quitter le groupe "INF DLM-B 2019".

## 5.2 Résultats

En Annexe ?

## 6 Améliorations

- Revoir le système de notification
- système d'abonnement
-

## 7 Conclusion

## 8 Annexes

- Installation et configuration du serveur
- Configuration de l'environnement de travail
- Planning
- Journal de travail

BAILLY Nicolas, 2020. *Laravel Sanctum Explained : SPA Authentication* [en ligne]. 2020. [Consulté le 20 juillet 2021]. Disponible à l'adresse : <https://dev.to/nicolus/laravel-sanctum-explained-spa-authentication-45g1>

CHRISTIANDY Eddy, 2019. *Simple Laravel + Vue + Laravel Mix + Firebase Notification (PWA, Offline)* [en ligne]. 2019. [Consulté le 20 avril 2021]. Disponible à l'adresse : <https://gist.github.com/echr/a1195e083e6ff4f980698fa06ebfec3>

DEGGES Randall, 2018. *Please Stop Using Local Storage* [en ligne]. 2018. Disponible à l'adresse : <https://dev.to/rdegges/please-stop-using-local-storage-1i04>

DR. DAVIS BOWMAN Jennifer, 2016. *Why Students Don't Do Their Homework—And What You Can Do About It* [en ligne]. 2016. [Consulté le mars 2021]. Disponible à l'adresse : <https://www.teachthought.com/pedagogy/why-students-dont-do-their-homework-and-what-you-can-do-about-it/>

DUTTA Roger, 2020-04-01. *Six reasons why students fail to complete their homework* [en ligne]. 2020-04-01. [Consulté le mars 2021]. Disponible à l'adresse : <https://www.academicgates.com/blog/six-reasons-why-students-fail-to-complete-their-homework/43/view>

EL-DIN Mohamed Alaa, 2020-03-04. *Laravel FCM (Firebase Cloud Messaging) Notification Channel* [en ligne]. 2020-03-04. Disponible à l'adresse : <https://laravel-notification-channels.com/fcm>

GOOGLE, 2021. *Firebase Cloud Messaging* [en ligne]. 2021. Disponible à l'adresse : <https://firebase.google.com/docs/cloud-messaging>

KATHURIA Pulkrit, 2019. *Mysql vs Postgresql performance test with Laravel API for simple Eloquent queries on 1 million items* [en ligne]. 2019. [Consulté le juillet 2021]. Disponible à l'adresse : <https://medium.com/web-developer/mysql-vs-postgresql-performance-test-with-laravel-api-for-simple-eloquent-queries-on-1-million-6e0e6f1005b8>

*méthode de MoSCoW*, 2021. [en ligne]. 15 février 2021. [Consulté le 6 mars 2021]. Disponible à l'adresse : [https://fr.wikipedia.org/wiki/M%C3%A9thode\\_MoSCoW](https://fr.wikipedia.org/wiki/M%C3%A9thode_MoSCoW)

TAF Fima, 2021-03-08. *How to add FCM (Firebase Cloud Messaging) to VueJS* [en ligne]. 2021-03-08. [Consulté le 20 avril 2021]. Disponible à l'adresse : <https://dev.to/vbanditv/how-to-add-fcm-firebase-cloud-messaging-to-vuejs-37np>

WENSHU Li, RICHARD Bennett, TAIMI Olsen et RACHEL McCord Ellestad, 2018. *Engage Engineering Students In Homework : Attribution Of Low Completion And Suggestions For Interventions*. [en ligne]. juin 2018. DOI 10.19030/ajee.v9i1.10186<sup>11</sup>. Disponible à l'adresse : [https://www.researchgate.net/publication/326191774\\_Engage\\_Engineering\\_Students\\_In\\_Homework\\_Attribution\\_Of\\_Low\\_Completion\\_And\\_Suggestions\\_For\\_Interventions](https://www.researchgate.net/publication/326191774_Engage_Engineering_Students_In_Homework_Attribution_Of_Low_Completion_And_Suggestions_For_Interventions)

<sup>11</sup> <https://doi.org/10.19030/ajee.v9i1.10186>